

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**EXTENSIBLE MARKUP LANGUAGE (XML) BASED
ANALYSIS AND COMPARISON OF HETEROGENEOUS
DATABASES**

by

Robert F. Halle

June 2001

Thesis Advisor:
Second Reader:

Valdis Berzins
Paul Young

Approved for public release; distribution is unlimited.

20010910 110

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
June 2001

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE

Extensible Markup Language (XML) Based Analysis And Comparison Of Heterogeneous Databases

5. FUNDING NUMBERS

6. AUTHOR(S)
Halle, Robert F.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING
ORGANIZATION REPORT
NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING /
MONITORING
AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release, distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

This thesis describes an Extensible Markup Language (XML) based analysis and comparison method that could be used to identify equivalent components of heterogeneous databases. In the Department of Defense there currently exist multiple databases required to support command and control of some portion of the battlefield force. Interoperability between forces will become crucial as the force structure continues to be reduced. This interoperability will be facilitated through the integration of these command and control databases into a singular joint database or by developing inter-communication schemas to support inter-database communications. The first step in either of these alternatives is the identification of equivalent components among the multiple databases.

This thesis describes how XML can be used to facilitate the process of analyzing and comparing multiple databases. Each step of the process is described in detail accompanied by explanations of the XML tools/resources required to execute the step and rationale of why the step is necessary. Detailed graphics and examples are employed to simplify and justify the step by step explanations. The JavaScript code developed as part of the research to execute the XML based analysis is included. This thesis concludes with discussions of the overall value of this XML based analysis and comparison process and of potential future work that could be pursued to further exploit this XML analysis and comparison method.

14. SUBJECT TERMS

Extensible Markup Language, XML Analysis, Heterogeneous Databases, Database Comparison, Database Analysis, C4I

15. NUMBER OF
PAGES
154

16. PRICE CODE

17. SECURITY CLASSIFICATION OF
REPORT
Unclassified

18. SECURITY CLASSIFICATION OF
THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF
ABSTRACT
Unclassified

20. LIMITATION
OF ABSTRACT
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

Approved for public release; distribution is unlimited

**EXTENSIBLE MARKUP LANGUAGE (XML) BASED ANALYSIS AND
COMPARISON OF HETEROGENEOUS DATABASES**

Robert F. Halle

B.S.E.E., University of Michigan, 1981


Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

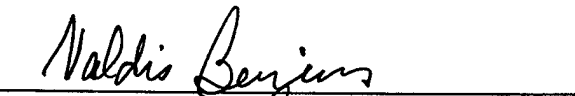
from the


**NAVAL POSTGRADUATE SCHOOL
June 2001**

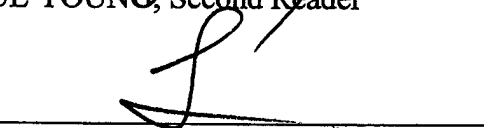
Author:


ROBERT F. HALLE

Approved by:


VALDIS BERZINS, Thesis Advisor


PAUL YOUNG, Second Reader


Luqi, Chair
Software Engineering

ABSTRACT

This thesis describes an Extensible Markup Language (XML) based analysis and comparison method that could be used to identify equivalent components of heterogeneous databases. In the Department of Defense there currently exist multiple databases required to support command and control of some portion of the battlefield force. Interoperability between forces will become crucial as the force structure continues to be reduced. This interoperability will be facilitated through the integration of these command and control databases into a singular joint database or by developing inter-communication schemas to support inter-database communications. The first step in either of these alternatives is the identification of equivalent components among the multiple databases.

This thesis describes how XML can be used to facilitate the process of analyzing and comparing multiple databases. Each step of the process is described in detail accompanied by explanations of the XML tools/resources required to execute the step and rationale of why the step is necessary. Detailed graphics and examples are employed to simplify and justify the step by step explanations. The JavaScript code developed as part of the research to execute the XML based analysis is included. This thesis concludes with discussions of the overall value of this XML based analysis and comparison process and of potential future work that could be pursued to further exploit this XML analysis and comparison method.

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	RESEARCH OVERVIEW	1
B.	C4I BACKGROUND.....	1
C.	OBJECTIVE OF THESIS:	3
D.	WHAT IS XML?	6
E.	SCOPE OF RESEARCH:.....	11
F.	LIMITATIONS IMPACTING RESEARCH:	12
G.	RESEARCH ASSUMPTIONS	13
H.	ORGANIZATION.....	14
II.	DATA AND PHYSICAL SCHEMA OF DATABASES	15
A.	JOINT COMMON DATABASE	15
B.	JCDB PRODUCTS	16
C.	MODERNIZED INTEGRATED DATABASE.....	16
D.	MIDB PRODUCTS	17
III.	SURVEY AND ASSESSMENT OF PREVIOUS WORK.....	19
A.	BACKGROUND.....	19
B.	DATABASES	19
1.	<i>OODBMS</i>	20
2.	<i>RDBMS</i>	20
3.	<i>RDBMS/OODBMS Advantages and Disadvantages</i>	21
C.	XML AND DATABASES	21
1.	<i>Similarities</i>	22
2.	<i>Differences</i>	23
3.	<i>Conclusion</i>	23
D.	COMMERCIAL DATABASE XML EFFORTS.....	24
1.	<i>Oracle</i>	25
2.	<i>Sybase</i>	27
3.	<i>Informix</i>	28
4.	<i>Summary</i>	28
E.	DEPARTMENT OF DEFENSE AND XML	29
1.	<i>What is a Namespace?</i>	30
2.	<i>DIICOE Namespace Registry</i>	34
3.	<i>XML-MTF</i>	36
4.	<i>Problems with USMTF</i>	38
5.	<i>XML-USMTF Mapping</i>	39
6.	<i>Other DoD XML Messaging Efforts:</i>	41
7.	<i>XML Mapping Caveat</i>	43
F.	PREVIOUS DATABASE ANALYSES METHODS.....	44
1.	<i>Research Preparation</i>	44
2.	<i>Roadblocks Encountered During Research</i>	45
3.	<i>An Opportunity for a New Analysis Method</i>	47
IV.	XML BASED ANALYSIS AND COMPARISON METHOD DESCRIPTION	49
A.	INTRODUCTION TO PROCESS	49
1.	<i>Focus of Process</i>	49
2.	<i>Database Analysis Aspects Not Covered</i>	50

B.	DATABASE REVIEW STEP	51
1.	JCDB/MIDB Comparison.....	54
2.	JCDB View	55
3.	MIDB View	56
4.	Conclusion.....	56
C.	DATABASE CONVERSION TO XML.....	56
1.	Database to XML Translation	57
2.	XML Translation Alternative.....	59
3.	JCDB and MIDB to XML Translations	59
4.	Conclusion.....	60
D.	ENTITY/ATTRIBUTE ANALYSIS AND COMPARISON.....	61
1.	Introduction	61
2.	Process Description.....	62
3.	Conclusion.....	62
E.	HIERARCHICAL EXAMINATION.....	63
1.	What is a DOM?	64
2.	Process Description Introduction.....	74
3.	Process Description Synopsis.....	76
4.	Analysis Process Description	76
5.	Summary of Analysis Process.....	87
F.	JCDB AND MIDB VIEW ANALYSIS	87
1.	Analysis Component Review.....	87
2.	JCDB Analysis.....	89
3.	MIDB Analysis.....	91
G.	FINAL STEP - COMPARISON OF ANALYSIS RESULTS	92
1.	Comparison Example	94
2.	Comparison Summary	97
H.	CHAPTER CONCLUSION	97
1.	Database Review Summary	98
2.	Database Conversion to XML Summary	98
3.	Entity/Attribute Analysis and Comparison Summary	99
4.	Hierarchical Examination Summary	99
5.	Comparison of Analysis Results Summary	101
6.	Research Limitations	101
7.	Example Crosswalk	103
8.	Commercial Application of Process	104
9.	Putting This Research Into Practice.....	104
V.	CONCLUSION AND FINAL RESEARCH POSSIBILITIES	106
A.	CONCLUSION.....	106
B.	FUTURE WORK RESEARCH POSSIBILITIES	107
APPENDIX A - JCDB DATABASE VIEW		109
APPENDIX B - MIDB DATABASE VIEW		113
APPENDIX C - JCDB XML DOCUMENT		117
APPENDIX D - MIDB XML DOCUMENT		119

APPENDIX E - ANALYSIS AND MANIPULATION CODE	123
APPENDIX F - JCDB ANALYSIS OUTPUT	129
APPENDIX G - MIDB ANALYSIS OUTPUT	131
GLOSSARY	135
LIST OF REFERENCES	137
INITIAL DISTRIBUTION LIST	139

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENT

This author would like to thank several people who supported and facilitated the writing of this thesis:

To Dr. Valdis Berzins and CAPT Paul Young for their guidance, encouragement, and most of all their patience during this research effort.

To my parents who provided encouragement and put up with my sometimes irritable moods when I visited.

To my supervisors during the periods I pursued my Masters. Mr. E. Robert DeGroot and LTC Ron Bokoch (Ret.) who always managed to provide me the freedom from my workload and travel as my studies required.

And finally, to the three other Master's candidates who are now struggling to complete their theses as I am: In Fall 1998, we four started out in a group of about 20, now it's down to just us. We were the only ones who went to all the long classes after working 9 or 10 hours days and completed all the coursework that was expected of us. It was a pleasure and honor to learn with you. I couldn't think of any other three people I would have rather experienced this with.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. RESEARCH OVERVIEW

The reason for this research is to examine methods that could improve interoperability between independently designed Command, Control, Communications, Computers, Intelligence (C4I) Systems. Each of these C4I systems employs a database to control and maintain its C4I information. The means to facilitate the interoperability between these different C4I systems will be by exchanging the data from their individual databases. The first step towards this data exchange will be to determine what parts of the individual databases are similar.

This thesis describes a method that can identify the similarities between C4I databases. It will employ Extensible Markup Language (XML) as a means to analyze these C4I databases and to extract portions from each for closer examination and comparison. The entire analysis and comparison process will be described in detail and executed using actual C4I databases.

B. C4I BACKGROUND

Command, Control, Communications, Computers, and Intelligence (C4I) systems have been developed and continue to be developed to support numerous and very diverse

military capabilities. These capabilities include mission planning, battlefield command and control, logistics management to name just a few. Each of these C4I systems retains large and complex databases that store the data necessary to execute its mission objectives. For example, the Global Command and Control System (GCCS) Integrated, Imagery and Intelligence (I³) utilizes the Modernized Intelligence Database (MIDB) to store weapons systems characteristics and national/tactical imagery to provide operational commanders enhanced situational awareness. The Advanced Field Artillery Tactical Data System (AFATDS) employs the AFATDS Database to retain the data required to support fire support planning, execution, movement, and support.

Most C4I systems, including the ones described above, are often called "legacy" systems. This is because they were developed several years ago to support very specific requirements. These legacy systems continue to be refined over the years to expand existing functionality and to incorporate new capabilities. As the amount and value of data assembled and employed by these legacy systems grew, it became apparent that the sharing of this data between multiple C4I systems would enhance combined arms management and increase the overall effectiveness of force. To support this data sharing, joint databases are being developed that

can interface with the legacy database. One example of a joint database is the Joint Common Database (JCDB).

The JCDB supports the Army Battlefield Command System (ABCS) by providing consolidated data from multiple Army systems to support development of a Common Tactical Picture for Army battlefield commanders. This consolidated data is also used to enhance the capabilities of the legacy systems by sharing the information awareness of each legacy system with the others.

C. OBJECTIVE OF THESIS:

The objective of this research is to identify a method (or methods) that can help distinguish and identify common data elements and physical schemas between dissimilar C4I databases. Extensible Markup Language (XML) is employed wherever possible to facilitate this identification task. Problems related to this identification task include:

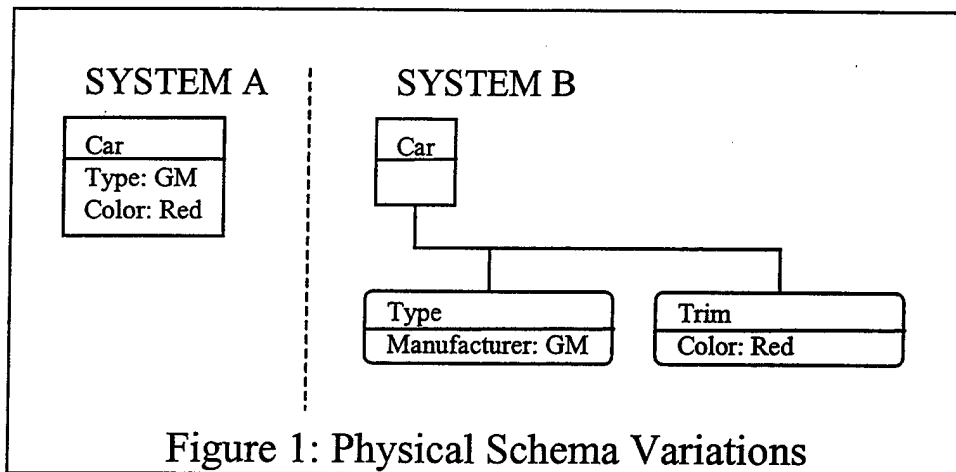
- Databases are extremely large: The JCDB consists of 526 tables including 315 look-up or reference set tables that are the data provider library to columns in the primary tables. There are a total of 1257 columns in the JCDB, of which 1147 are unique. Others appear in more than one table [JDD00]. To graphically display the physical schema using entity relationship diagrams would take over 350 pages.

- There are many databases: The many and often dissimilar databases (like the ones identified earlier)

continue to evolve and change to incorporate new data to support new functionality and remove unneeded data that supported antiquated functionality.

- Database Terminology Variance: Subject matter experts defined terminology that specifically related to C4I system functionality as each database was developed independently. This terminology variance can impact the comparison of dissimilar databases. The terms "Tank" and "Armored Vehicle" can be used to identify a heavily armored, mobile, direct fire weapons system. The term "Tank" can also be used to describe a water storage tank.

- Physical Schema Variance: Physical schemas can vary from database to database even though they describe the same thing. For example, the following figure shows two different representations of the same object. System A describes everything in one object. System B uses the attributes to describe the object in different lower level objects. As a whole, physical schema A and physical schema B describe the same object.



- Required Human Intervention: Subject Matter Experts (SMEs) will always have to be consulted when examining heterogeneous databases. Figure 1 shows multiple terms and physical schemas can be used to describe the same object. SMEs will always have to be consulted when comparing complex objects to make the final determination of whether the identified common data elements are truly common and whether the associated physical schemas are common.

The problems identified above mandated that the objectives of this research project be refined to include the following:

- Need for a method that can simplify the search and comparison of multiple and very large databases.

- The method must provide SMEs the information necessary to make the final determination of what is common and what is not common. This reduces work by showing the SME only the parts that are likely to be related.

D. WHAT IS XML?

This research has been carried out in the context of the markup language, XML. But what is a markup language? Historically, the term markup originated as part of the document publishing process. Documents would be "marked up" by authors and editors to reflect style and format instructions for the printers on how the document should be printed. Over time these markup comments evolved into markup languages. The goal of these markup languages was to convey specific information on the text using a unique format so that it wouldn't be confused with the text itself.

In 1969, Ed Mosher, Ray Lorie, and Charles F. Goldfarb of IBM Research developed the Generalized Markup Language (GML) [And00]. GML was the first markup language developed to support modern electronic publishing needs. It was a meta-language that was used to describe other languages, their grammars, and vocabularies.

GML eventually became the Standardized Generalized Markup Language (SGML). In 1986, SGML was adopted as an international data storage and exchange standard by the International Organization for Standardization (ISO) designated ISO8879.

SGML's goal was to define descriptions of the structure and content of different types of electronic documents. The problem with SGML is that it is extremely complicated and

was only used by those who do large amounts of publishing like newspaper companies and the publishers of technical information. This complexity has increased over many years as more and more is added to the specification to support evolving publishing requirements. This growing complexity has limited SGML use to those companies/ organizations that could absorb the high cost of implementing and maintaining SGML proficiency.

With the advent of the internet/web, Hypertext Markup Language (HTML) was developed from SGML as an easily understandable markup language. The primary objective for HTML was that it had to be a simple markup language that a user could use to describe a document structure as well as the role each part played, regardless of how it looked on the monitor. The benefit of using HTML is that it is easy to learn, it is supported by the two primary browsers (Explorer and Netscape), and most of all it's cheap. This last benefit is the primary reason for the explosion of HTML's use: Anybody can use HTML.

A problem with HTML is that it doesn't provide the ability to easily modify the size/structure of the document to the size of the screen. As described earlier, that is one of primary purposes for a markup language is to provide structure to a document for publishing. In HTML's case, it's the publishing of information on the web to any platform that can display web pages. For standard personal

computers, the display of this information is done extremely well. But HTML has a more difficult time displaying that information on mobile electronic devices like Palm Pilots, pagers, and cell phones that are interfaced with the internet.

Another problem with HTML is that it locks the data with the presentation format. Once the data is built into an HTML page, it is no longer easily accessible to the user and cannot easily be displayed in a different format or be processed further by other programs.

These two problems, along with many others, were why XML was developed and why it has rapidly gained in popularity.

In 1996, the World Wide Web Consortium (W3C) began to develop an extensible markup language that would combine the flexibility and power of SGML with the acceptance of HTML [And00]. This was the start of the Extensible Markup Language (XML).

As the starting point for XML's creation, W3C defined 10 design goals [XMLR1.0]:

1. XML shall be straightforwardly usable over the internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.

4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

But what is XML? Simply put, XML is an open family of markup language with which you can design ways of describing data, usually for storage, transmission, or processing by a program. XML is a meta-language for describing markup languages thus providing the facility for a person to define, or extend, his/her own descriptive terms for the data and the structural relationships between the descriptive terms. This is what the word "Extensible" in Extensible Markup Language means and is the foundation of XML. The following describes XML functionality:

- XML is extensible: An XML document can be easily developed and understood. Data elements and attributes are defined to personalize the XML document to meet any specific requirement(s).

- XML relates well to relational databases: XML creates and maintains a hierarchical data structure. Relational database systems use relational models to associate data entities/tables to simplify data sorting, searching, and retrieval. Although these two data structure approaches are very different, they both provide the means to create hierarchical structures that can be maintained and shared.

- XML is not tied to any particular context: There is no requirement for any particular programming languages, operating system, or computer platform to build and process XML documents. All computer platforms have simple text editors that can be used to develop XML documents.

- XML is self-describing: An XML document should be easy to read. This understandability results from the process of defining the data elements and related hierarchy based on the designer's own "common sense" perspective of the data.

- XML provides opportunity for development of standardized data representations: The transfer of data between different database systems developed by different manufacturers using different operating systems has been a complicated process. XML has the ability to support development of independent data formats that multiple database systems can understand.

E. SCOPE OF RESEARCH:

The Joint Battle Center (JBC) and the Naval Postgraduate School (NPS) defined the scope of this research. The Joint Battle Management Initiative Assessment Plan [JBMI00] describes the NPS led XML Schema Investigation research to execute the following:

- Determine methods for assuring scalability of solution to legacy and migration C4I. This requires a method(s) that can support analysis and comparisons of both legacy and evolving common databases like JCDB.

- Determine what parts of a legacy system view could be materialized from previous shared schemas. This requires the identified method(s) to be able to identify common elements and physical schemas.

- Determine how to materialize those parts relevant to such an assessment. This requires the identified common data elements and schemas be integrated into a global common schema.

This thesis describes the research associated with the examination of previously developed analysis methods that support different aspects of the defined tasks. Then a new and original XML-based database analysis and comparison method, developed as part of the research effort, is described in detail. This method is demonstrated to show how it supports the described JBC/NPS tasks.

F. LIMITATIONS IMPACTING RESEARCH:

The original research objectives (described earlier) for this thesis effort involved detailed comparisons of the JCDB, AFATDS, MIDB, and GCCS Track Database Manager Database (TDBM). As directed by NPS, the particular task to be investigated in this research effort focused on the analysis of the JCDB and AFATDS databases.

Unfortunately, the AFATDS database dictionary and entity relationship diagrams proved to be unattainable in sufficient time to support this research effort. After several months of effort by several organizations and individuals (including myself) information relating to the AFATDS database could not be acquired. The AFATDS database is a closely held document that would not be released to support a master's level research project. The unavailability of this critical component forced a modification to the original research objectives from the examination of the AFATDS to the examination of the MIDB. The MIDB Data Dictionary provided sufficient detail to support the development of entity relationship diagrams that represented certain views of the MIDB. The developed views of the MIDB were then compared against JCDB views extracted from the provided JCDB Data Dictionary and Entity Relationship Diagrams.

The sheer size of JCDB and MIDB presented a significant challenge when executing an analysis of the two databases. The specific database application software was required to process these large databases. Since the application software could not be acquired in time to support this research effort, smaller views of particular portions of the two databases were chosen for examination.

Another limitation faced during this research effort was that XML is still evolving, in some cases evolving rapidly. Some of the necessary recommendations are still in draft form. It is expected that within one year all necessary XML Recommendations (to be discussed later) will be finalized and approved. It will then take the commercial sector time to develop and field software that incorporates these XML based capabilities. However, in some cases, the commercial sector is already producing software that incorporates capabilities defined in the draft recommendations. When required, the XML based analysis process to be described in this thesis had to employ XML based capabilities that are still undergoing review.

G. RESEARCH ASSUMPTIONS

Assumption #1: The XML related draft recommendations are stable and will be finalized in virtually the same form and content.

Assumption #2: Current XML commercial sector developments will continue at the same rapid pace once all XML related recommendations are finalized and approved.

H. ORGANIZATION

This thesis is organized into the following chapters:

- Chapter II describes the databases to be examined in this research effort.
- Chapter III provides a review of previous work in this area. This includes descriptions of other applicable examination methods, ongoing Department of Defense XML efforts, and a rationalization of why a new XML based analysis method is required.
- Chapter IV presents the XML based analysis method developed during this research effort. The overall analysis process is described. An execution of the developed method with an analysis and comparison of JCDB and MIDB views will follow the method description.
- Chapter V presents the conclusions and recommendations of future work derived from this research effort.
- Following Chapter V are several appendices containing the materials used to support this research effort. Also included is the code developed that executes this XML based analysis. The appendices also contain the thesis glossary and references used in this research effort.

II. DATA AND PHYSICAL SCHEMA OF DATABASES

The two databases to be analyzed in this research effort are the JCDB and MIDB. Both are considered high level common databases that incorporate data from several systems for use by commanders who require information from many sources to execute battle management.

A. JOINT COMMON DATABASE

The JCDB is a key component of the U.S. Army's efforts to employ common software and data across command and control (C2) systems [JCDB99]. The JCDB resides on the Army Battlefield Command System (ABCS) and provides the data necessary to support the common applications that build the Common Tactical Picture (CTP). Some of the information displayed as part of the CTP is:

- Friendly and enemy locations, activities, strength, status, estimated and current capability.
- Tracking of resources.
- Tracking of materiel locations, status, and quantities.
- Mapping of ground and air control measures.
- Mapping of facilities.
- Target nomination, engagement, and damage assessment.

- Evaluation and verification of reported information.
- Development of operational orders and operational plans.

B. JCDB PRODUCTS

The JCDB products available to support this research effort were the Joint Data Model and the JDD. The Joint Data Model is a logical data model that displays the entity relationships of the JCDB entities and attributes. The JDD is a data dictionary that provides data entity names, definitions, datatype and domain values/enumerated types for the entities.

C. MODERNIZED INTEGRATED DATABASE

The GCCS I3 system objectives are to provide accurate, user friendly, and immediately accessible Integrated Imagery and Intelligence (I3) capabilities to support the warfighter. The GCCS I3 provides commanders with situational awareness and track management with a standard set of linked tools that maximizes commonality across the tactical, theater, and national communities [I3BR]

The GCCS I3 provides access to the information maintained in the MIDB. This information includes:

- National and Theater-level intelligence on facilities.
- Order-of-Battle

- Information on equipment and targets (including target assessments)
- Derived intelligence entered by tactical intelligence assets

This information is used to provide operational commanders and intelligence analysts quick access to intelligence and imagery through the Common Operational Picture (COP).

D. MIDB PRODUCTS

The only MIDB product available to support this research effort was the Modernized Integrated Database-Database Design Document (MIDBD3). The MIDBD3 is a data dictionary that provides data entity names, definitions, datatype and domain values/enumerated types for the entities.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SURVEY AND ASSESSMENT OF PREVIOUS WORK

A. BACKGROUND

With the focus of this research effort to locate and analyze commonality between heterogeneous databases it is important to understand the components that will make up the identification process. The databases types, XML-database relationships, and XML-information sharing need to be understood before delving into an evaluation of previous methods.

B. DATABASES

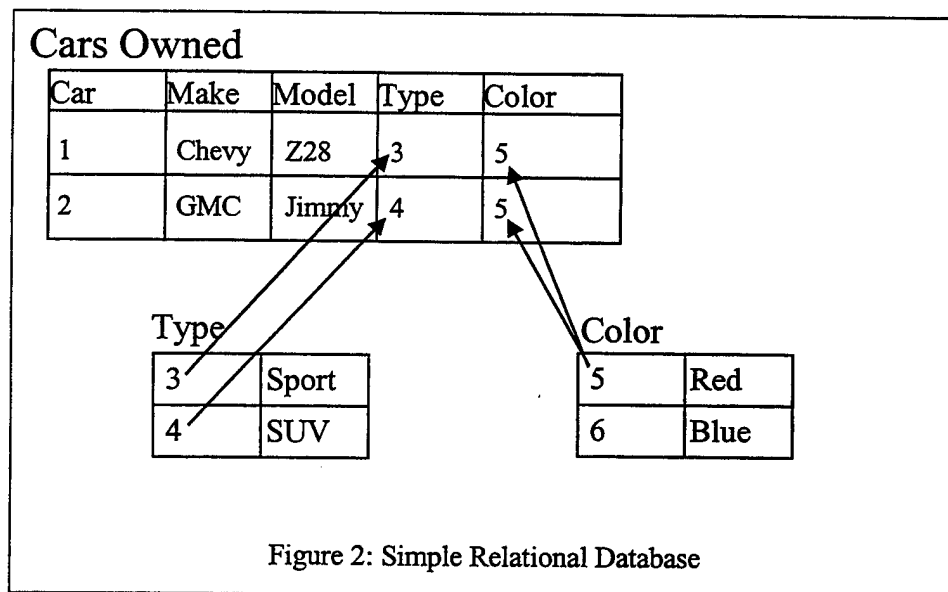
There are two types of database management systems available to represent data. They are the Object Oriented Database Management System (OODBMS) and the Relational Database Management System (RDBMS). Each provides capabilities that support unique database storage and manipulation capabilities. Object Oriented Database Management Systems provide the capability to deal with objects that have complex relationships and depth [AND00]. Relational Database Management Systems provide the capability to model many real world problems and provide more thorough and rapid manipulation of the data [AND00].

1. OODBMS

OODBMSs provide the capability to build objects and relate those objects to other objects. A complex object interrelationship can quickly develop when the database contains several levels of objects. The OODBMS provides the capability to build these complex databases and to manipulate them. This manipulation includes the ability to add, delete, and modify nodes anywhere in the object-oriented database (OODB) without impacting the rest of the database. OODBMSs also provide the standard set of facilities to search and retrieve data from the database.

2. RDBMS

RDBMSs use tables comprised of rows and columns to store and relate data. The row and column headings provide the means to define the data. The simple example of a RDBMS shown in Figure 2 provides information on cars. The tables



provide a simple representation of two cars. The top-level "Cars Owned" table provides overall data relating to the cars. The "Type" and "Color" Tables are joined to the Cars Owned Table. This joining mechanism defines how the different tables are related. A relational database is built through these relationships between multiple tables.

3. RDBMS/OODBMS Advantages and Disadvantages

It is the RDBMS's efficiency, simplicity, and ability to support most database storage and manipulation problems that makes it more popular and more widely used than OODBMSs. This stems from the fact that RDBMSs have been in existence longer and are more mature than OODBMSs. The primary disadvantage with RDBMSs is that when modeling extremely complex relationships the RDBMS's efficiency can be affected. OODBMSs can better represent these complex database relationships.

C. XML AND DATABASES

Previous sections of this thesis describe XML and databases. This section will describe the similarities and differences between XML and databases. In most cases though XML and databases (and related DBMSs) do provide similar data search and manipulation capabilities, it's just the extent to which they employ those capabilities that defines their similarities and differences. In general, descriptions of database technologies apply to both RDBMSs

and OODBMSs. If the description specifically relates to one or the other, it is specifically identified.

1. Similarities

Both XML and database technologies maintain data in hierarchical, parent/child relationships. XML and OODBMSs can maintain extremely complex, very deep relationships. As discussed earlier, RDBMSs do not efficiently process these complex relationships as well.

XML and database technologies provide the ability to search and manipulate data. Since the DBMSs provide this capability through their unique internal schemas and query languages they are very efficient in their execution. XML provides the designer/developer a great amount of flexibility in the design of their documents through the use of internal or external Document Type Definitions (DTDs) or Schemas, namespace definitions, etc. The XML document can be just about any size and can retain an unknown level of complexity. To cope with this flexibility of design, XML employs open-ended languages and Application Programming Interfaces (APIs) like XQuery, XPointer, XLink, and Document Object Model (DOM) APIs to support this flexibility of design. Certain speed of service inefficiencies result from having to account for these open designs.

2. Differences

As described earlier, DBMSs provide a database manipulation efficiency that is not found in XML. The DBMS's efficiency stems from their strict definition of data structure and the data search and manipulation capabilities specifically tailored to that data structure. These efficiencies of DBMSs and XML inefficiencies reflect the fact that XML is not a DBMS.

XML provides a single structured view of the data as defined by the DTD or XML Schema. Rearranging of the data will require a corresponding modification to the DTD or XML Schema. DBMSs retain the capabilities to easily modify the structure of the database.

The XML document is basically a text-formatted document that is independent of any particular platform or software application. DBMSs store their databases in DBMS specific formats. The DBMSs must be operated from very specific computing platforms with specific software applications.

XML is not restricted by any document structure. It can reflect any structure the designer desires. Databases developed in DBMSs must adhere to specific database structures.

3. Conclusion

This XML - database comparison demonstrates that though there are differences between the two, the common thread

between them is the data. Each maintains and manipulates the data in their own way. Recognizing the value of XML, most DBMS manufacturers are beginning to integrate XML resources and capabilities into their software. These XML resources and capabilities will be described in detail in the following section. An example of one XML capability being integrated is an XML translator that supports the translation from a specific database format into an XML document. This allows the DBMS to take advantage of web based functionality associated with XML.

D. COMMERCIAL DATABASE XML EFFORTS

The primary advantage for DBMS manufacturers for integrating XML capabilities into their products is to support web transmission and manipulation of their databases. Another advantage is that the use of XML simplifies inter-database transmission. In the past, communications between different brands of database products required the development of complex translators. These translators translate the database from one database format to another. While this solution worked, it also required the translator to undergo costly rebuilds each time one or the other DBMSs changed. XML provides the means to support translation from one database format to XML and then from XML to another database format.

Other XML capabilities being integrated include XML parsers, XML storage, and XML document queries. This thesis will briefly examine the XML capabilities being integrated into the Oracle, Sybase, and Informix DBMSs. The XML related information presented comes from each of the manufacturers web sites.

1. Oracle

Of all the DBMS manufacturers, it appears Oracle is investing in XML the heaviest. They have developed the Oracle Internet Platform that provides integrated support of internet standards, including XML and JAVA [ORCWS]. Oracle 8i contains a built-in JAVA Virtual Machine that can execute Oracle's XML based components.

Oracle's interMedia Text can be used to perform searches on XML documents stored in Oracle 8i. The foundation for these searches are simple textual matching algorithms that can be used on an individual XML document or on the entire database that contains the documents.

Oracle provides the capability to store the XML document in its original XML structure. This capability is advantageous if the document does not have to be updated often or if it has to be transmitted as a whole. Otherwise, Oracle provides the capability to store the XML document as data in the Oracle database format. This storage

alternative is beneficial if the XML document is undergoing a significant amount of updates to the data.

Oracle stores the XML document in its database format by capturing the attributes of the data elements in a relational table and the objects are defined to convey the XML document structure. The XML Structured Query Language (SQL) utility provides the means to store and retrieve the document from the database. Once stored in the database, the data can be updated, queried, rearranged, reformatted, and extracted as required.

Oracle Views can be used to store an object "on the fly" by combining XML data stored in many ways. In effect the XML document is stored twice in Oracle with each linked to the other. This approach provides the ability to access the entire document along with the data in the objects defined in the database. The XML SQL provides the ability to extract the assembled data as a single XML document.

Oracle provides several XML parsers to support JAVA, C, C++, and PL/SQL applications. The parsers also provide support to the Document Object Model, Simple API for XML (SAX) interfaces, XML Namespaces, and Extensible Stylesheet Language Transformation (XSLT). All of these parsers provide XML document-validating capabilities.

Oracle utilizes XML to support exchange of data between business applications. Oracle exploits the DTDs in the XML documents to identify the common data elements and structure

to support sharing of the data. To share data from one XML document to another, the structure and content of the data being shared has to compare favorably between the two XML DTDs. If the DTDs do not share common data elements or structure, the necessary portions of one or both of the documents are transformed using XSLT into a common format thus allowing one document to share data with the other.

In the future, Oracle plans on expanding its XML querying capabilities to enable not only identification of the individual data elements but also the logical view related to that data element. This will provide the capability to search for specific data elements and/or specific XML document structure.

2. Sybase

Sybase's Adaptive Server Enterprise 12.0 provides the ability to create, store, retrieve, and query XML documents. Based on the information provided in the Sybase web site [SYBWS], Sybase provides an XML parser to ensure all XML documents developed are valid. Once deemed valid, Sybase provides the capability to store XML documents in relational tables. The entire XML document can be stored as a whole or as native text in text or image columns. Any XML transactions can be mapped into new or existing relational tables. Sybase provides integrated textual search capabilities to query the document and to create XML

formatted query results for incorporation into XML documents.

The JAVA capabilities, provided by Sybase's Server Enterprise 12.0, can be used to incorporate any commercially available JAVA-based XML tools.

The capabilities provided by Sybase are comparable to those provided in Oracle. Perhaps the one advantage Oracle has over Sybase is that their products are more widely used in the commercial sector.

3. Informix

Informix intends on exploiting XML to support data sharing. The initiative that supports this ability is the Informix Internet Foundation 2000. This foundation consists of their Informix Dynamic Server 2000 and a series of tools to support Internet applications. The Informix Web Datablade Module is the first of the modules that has been created to support exploitation of XML [INFWS].

Informix provides the ability to create, store, and query XML documents in their database. It also supports the integration of XML documents with legacy data in the database.

4. Summary

Each of the DBMS described above provides the ability to store, search, manipulate, and extract XML documents from their databases. This ability to extract XML documents from

databases significantly simplifies the process to execute the XML based analysis method that will be described in subsequent sections.

E. DEPARTMENT OF DEFENSE AND XML

As with the commercial sector, the Department of Defense (DoD) has recognized the value of XML to aid in the interoperability between dissimilar systems. They also recognized the danger of several DoD organizations possibly developing their own unique XML representations that could hinder interoperability. As a result, the DoD has begun a concerted effort to intelligently control XML development and implementation as it relates to interoperability. One means of control is the use of a DoD-wide XML Namespace Registry to categorize and maintain common and reusable XML elements.

Draft guidance entitled "Guidance on the Use of Extensible Markup Language within DoD" [DISA00] was released on 29 Aug 00. The primary guidance states that the DoD will implement a common DoD registry of namespaces. All developers of XML documents will be required to review this repository for reusable tags, elements, and constructs before developing new ones. If new tags, elements, or constructs are developed it should be determined if they are reusable. If so, the developer should submit the reusable components to the registry.

1. What is a Namespace?

Namespaces are used in XML to ensure uniqueness of the XML elements. This ability to define uniqueness is important in XML since it is this uniqueness that will allow one element to be distinguished from another. The extensibility of XML allows the XML document developer to create any element desired. Problems can arise when an element could be interpreted in multiple ways. The following two examples demonstrate this problem:

Example A: Compare the following two XML documents adapted from example in [BOUR00]:

```
<?XML Version="1.0"?>
```

```
<Address>
```

```
    <Street>11 Mile</Street>
```

```
    <City>Warren</City>
```

```
    <State>Michigan</State>
```

```
    <Zip>48397</Zip>
```

```
</Address>
```

and

```
<?XML Version="1.0"?>
```

```
<PersonalID>
```

```
    <Address>XXX@fake-email.com</Address>
```

```
</PersonalID>
```

Example B: Also compare the following two XML documents:

```
<?XML Version="1.0"?>

<Tank>

    <Type>Heavy</Type>

    <Nomenclature>M1</Nomenclature>

</Tank>

and

<?XML Version="1.0"?>

<Tank>

    <Type>Water</Type>

    <Capacity>500 USGal</Capacity>

</Tank>
```

One of the greatest advantages in using XML is that it allows the XML developer to develop the XML elements that best suits their needs. This extensibility also can cause problems. The two previous examples show how like terms can define very different things. This is why namespaces are required.

But what is a namespace? "XML namespaces are collections of names, nothing more" [BOUR00]. Namespaces provide the ability to develop unique identifiers for XML elements. This can be demonstrated by reexamining the previous examples that now incorporate namespaces:

Example C: Adapted from example in [BOUR00]:

```

<addr:Address
xmlns:addr="http://www.????.com/addresses">

  <addr:Street>11 Mile</addr:Street>

  <addr:City>Warren</addr:City>

  <addr:State>Michigan</addr:State>

  <addr:Zip>48397</addr:Zip>
</addr:Address>

and

<serv:PersonalID
xmlns:serv="http://www.***.com/emailaddresses">

  <serv:Address>XXX@fake-
  email.com</serv:Address>

</serv:PersonalID>

```

Example D: Also compare that following two XML documents:

```

<ArmyVehicle:Tank
xmlns:ArmyVehicle="http://www.***.com/vehicles">

  <ArmyVehicle:Type>Heavy</ArmyVehicle:Type>

  <ArmyVehicle:Nomenclature>M1
  </ArmyVehicle:Nomenclature>

</ArmyVehicle:Tank>

and

<Storage:Tank xmlns
Storage="http://www.@@@.com/Storage">

  <Storage:Type>Water</Storage:Type>

  <Storage:Capacity>500
  USGal</Storage:Capacity>

</Storage:Tank>

```

The rationale for using namespaces is that it allows an XML developer to freely distribute their XML documents to others since the potential XML element conflicts will be eliminated. The use of namespaces simplifies the process of using XML to communicate data and in the case of the DoD, to support interoperability since each element will be unique and understandable per the namespace definition.

In Examples C and D the Unified Resource Identifier (URI) (shown in bold below) identifies the namespaces:

```
<addr:Address xmlns:addr="http://www.???.com/addresses">
```

URIs are used because they are a well-known system for creating unique identifiers [BOUR00]. These URIs are maintained by a singular owner/organization that has the responsibility of defining the namespace common elements.

Recognizing the need for namespace ownership, the DoD XML Guidance [DISA00] states the following: "Once approved, the namespace manager will exercise aggressive oversight of his namespace." and "...the namespace manager will...establish a registry and repository (R&R) of namespace specific elements and constructs." The Namespace Registry is a mechanism through which the relevant elements and constructs can be registered to coincide with a specific location that can be located through queries. The Namespace Repository is the location where the registry resides and

from which they can be located and retrieved. The top level XML Namespace R&R is maintained by Defense Information Systems Agency (DISA) with the individual owners/namespace managers responsible for their own particular namespace.

DISA's overall role in XML namespace management is part of their management of the Defense Information Infrastructure Common Operating Environment (DIICOE). The DIICOE is a data environment defined to support "...interoperability and software reuse in a secure, reliable, and global networked environment"[DISAWS]. The DIICOE's data service infrastructure employs "sets of shared schemas, data management and services, build-time and runtime tools, server development and operating procedures, and technical guidance for supporting COE-based mission applications". The goal for the DII is to migrate from many dissimilar data stores to a set of standardized COE compliant data services.

DISA is using SHARED Data Engineering (SHADE) as the DIICOE data emporium that maintains these COE compliant data services. Contained within the SHADE is the XML Namespace Registry.

2. DIICOE Namespace Registry

DISA's Namespace Registry standardizes a set of elements developed, coordinated, and approved in the COE community. The registry provides the user the ability to

search and retrieve these common elements. If a desired element (or set of elements/constructs) cannot be located, the user can submit a proposed element (or set of elements/constructs) to the "Community of Interest" (COI) [DISA00] for consideration of incorporation into the XML Registry. Currently there are 11 COI's:

COI	Owner
COE Enterprise	DISA-DIICOE Chief Engineer
Ground Operations	Army-PM for JCDB
General Military Intelligence	Defense intelligence Agency
Aerospace Operations	USAF-AF Common Data Environment Staff
Messages	DISA-Chair USMTF CCB
Track and Reports	Navy-Common Track Data Store Engineer
Geospatial & Imagery	NIMA-NIMA Engineer
METOC	Navy-Developer of Meteorological and Oceanographic Models
Combat Support	DISA-CCSS Engineer
Finance	DFAS
Personnel	DIMHRS

One of the XML COIs identified above is the Message COI. This COI primarily focuses on the namespaces identified to support ongoing XML-MTF effort led by the Air

Force. The COI provides a means to facilitate determination of tag names for the corresponding XML translations of the JCDB.

3. XML-MTF

The single largest XML effort ongoing in the DoD is the XML-United States Message Text Format (USMTF) initiative. This effort focuses on the capability to build XML translations of USMTF messages and USMTF translations of XML documents. USMTF is one of the message formats used to convey data to the JCDB and MIDB.

USMTF is a text (character) oriented message format used to support tactical and support communications. Currently there are over 350 different message types used. A primary objective for the USMTF program is to support the production of messages that can be read by humans and machines. In effect, USMTF is an artificial language that employs a controlled vocabulary. This vocabulary allows the user to develop messages that can be understood by both humans and machines. This understandability is important since the USMTF messages are used for inter-service and with allied communications. The vocabulary is comprised of words arranged in predetermined formats that convey specific information based on the location of the words and their meaning.

The syntax of the USMTF message defines how it is structured. The basic structure is comprised of message, sets, and fields. The natural language equivalents are words, sentences, and text. The vocabulary, of an MTF message consists of formats for the fields, sets, and message. The terms that complete these fields are represented by field contents, set format identifiers, and message test identifiers.

The structure, semantics, and syntax are defined in MIL-STD-6040. It is this standard that defines the USMTF schema. The following example displays how USMTF messages can be structured. The following example shows a USMTF columnar structured messages [AMP99]:

UNCLAS

```
EXER/OLIVE DRAB 99//  
MSGID/SITREP/AFOP-JT//  
REF/A/ORDER/CTG122.4/161500ZJAN1999/0101006//  
PERID/172000Z/TO:181800Z/ASOF:171800Z//  
MAP/1501/11/1/6-GSFS//  
HEADING/ENEMY//  
AMPN/LIGHT RESISTANCE, ENEMY CASUALTIES UNKNOWN//  
5EUNIT  
/DE/CY/ACTTYP/ENUNIT          /UNITLOC          /TIMPOS  
/01/RS/DEPLOY/345 MTR RFL DIV/RIDGELINE CHARLIE  /171200Z//  
HEADING/OWN SITUATION//  
BNDLINE/FEBA/50QRD99109920/50QRD99309940/50QRD99509960/50QRD  
99809908//
```

BT

#0009

4. Problems with USMTF

A major problem with USMTF messages is that they can not be uniformly exchanged between all C4I systems. There are several C4I systems fielded that don't use USMTF as the means to communicate. A common COTS based method of representing messages would improve interoperability between C4I systems.

A second problem is that USMTF is a government managed standard that has been in use for many years by all services and many allied commands. The USMTF related software has to be developed, fielded, and maintained at significant cost to the government. It was realized that there could be advantages in pursuing commercial-off-the-shelf (COTS) based alternatives for USMTF. Cost savings could be realized from adopting a COTS based alternative since a wider variety of systems might be able to recognize a COTS based version of USMTF messages.

Another problem with USMTF messages is that they can not be easily read [HOP99]. As can be seen from the USMTF example the message is not inherently readable unless you have a detailed understanding of the USMTF vocabulary.

A final problem is that USMTF messages can not be easily prepared and are subject to errors. Extensive use of MIL-STD-6040 and other references is required to prepare the messages.

5. XML-USMTF Mapping

In 1998, Mitre Corporation working for the Air Force began to investigate if XML could be employed to deal with the USMTF problems described previously. It was determined that XML could provide an alternative method to represent USMTF data and structure. It was also accompanied by other COTS/XML based resources/capabilities (like XSLT) that supported transformation of that data. The alternative, called "XML-MTF" provides several benefits [HOP99]:

- XML-MTF provides improved flexibility when displaying data. Extensible Stylesheet Language (XSL) can be used to reorganize and reformat the data to simplify understandability. In fact, innovative stylesheets could be created to ensure battle commanders could visualize a common operational picture no matter what system they were using.

- Software maintenance cost savings could be realized through the use of COTS based XML-MTF parsers. The COTS based parsers could run on COTS platforms and reduce reliance on costly legacy software and hardware.

- XML-MTF could utilize the same network transmission protocols as the World Wide Web. This would allow the XML-MTF documents to be transmitted over commercial networks as easily as HTML.

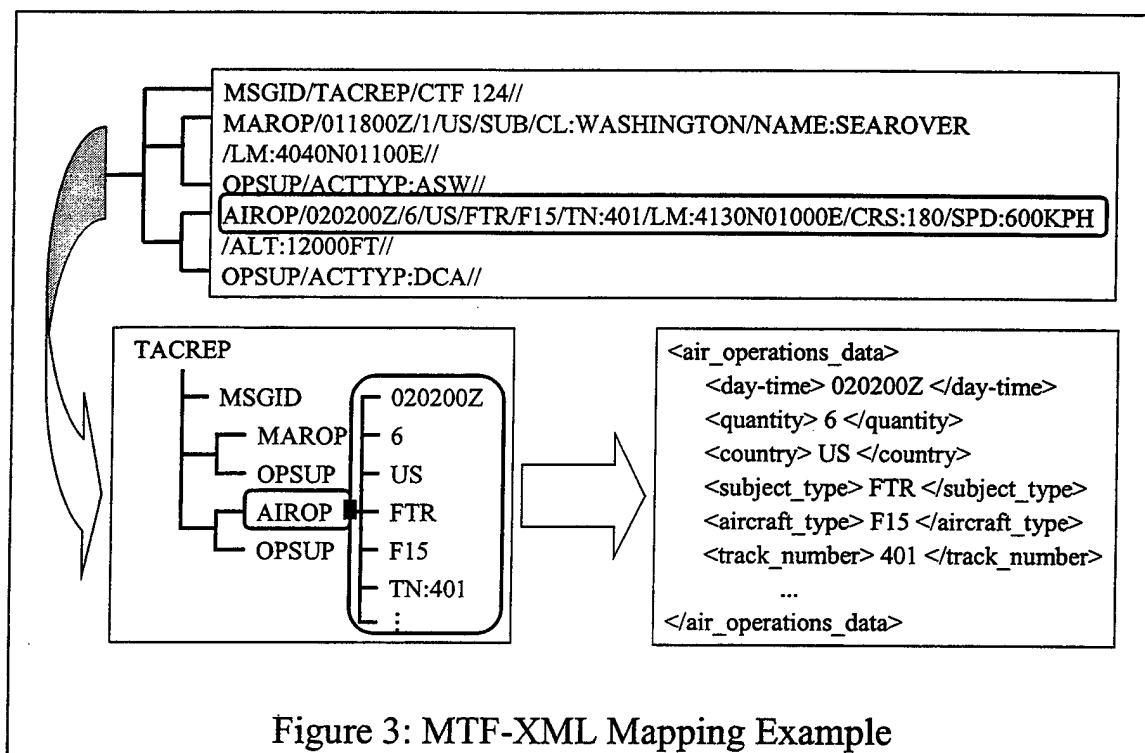
- XML-MTF could be used to simplify common data updates to dissimilar databases. A single database could be used to

"push" common data to the different databases. This would ensure the data was consistent throughout all the databases. The alternative requires individual development of database updates for each type of database (i.e., AFATDs database, MIDB, TDBM database, etc.).

XML-MTF Mapping is the primary product of the XML-MTF effort. XML-MTF Mapping consists of the definition of the formatting rules for the XML-MTF messages. These rules are included in Appendix A of MIL-STD-6040. The design goals guiding the XML-MTF Mapping effort are [MAP00]:

- XML-MTF shall be easy to read and understand.
- XML-MTF shall be designed to ensure widespread military adoption by accommodating current MTF standards.
- XML-MTF should be easy to construct from basic rules mapping to MTF formats.
- XML-MTF schemas should be easy to construct.
- Operations to XML-MTF messages should be resilient to schema change.
- XML-MTF shall draw as much as possible from industry standards.

The details of how XML-MTF Mapping is being developed can be found in the XML-MTF Mapping Third Public Working Draft [MAP00]. Below is an example mapping a portion of an MTF message to XML-MTF taken from an XML-MTF Update Brief [XMLMP00]:



This XML-MTF Mapping Effort has developed XML-MTF Translators to simplify the translation process. These translators translate from MTF to XML-MTF and from XML-MTF to MTF. The XML-MTF Mapping Effort continues to refine and expand the XML-MTF Mapping and is pursuing NATO adoption of this process.

Other XML-MTF mapping information can be found in Lieutenant Todd Ehrhardt's and Captain Brian Lyttle's thesis entitled "Interconnectivity Via a Consolidated Type Hierarchy and XML" [EHL01].

6. Other DoD XML Messaging Efforts:

Based on the success of the XML-MTF initiative, DoD has initiated the XML-OTG Mapping initiative. The full name

for OTG is Operational Specification for Over the Horizon Targeting Gold (OS-OTG). Like USMTF, OS-OTG is a character oriented messaging standard and is used primarily in naval communications. The primary differences between USMTF and OS-OTG are in the message structure, syntax, and rules.

The similarities between USMTF and OS-OTG allowed those executing the XML-OTG Mapping effort to leverage the previous XML-MTF Mapping. The first XML-OTG Mapping Working Draft, released 31 August 2000, bears a remarkable similarity to the XML-MTF Mapping Working Draft. In those areas where USMTF and OS-OTG are the same, the mapping approaches used in the XML-MTF Mapping were adopted in the XML-OTG Mapping Working Draft. The remainder of the XML-OTG Working Draft identifies the various differences between USMTF and OS-OTG and then identifies corresponding mapping approaches to resolve those differences.

Other messaging formats are now being examined to determine if they are candidates for XML mapping. The Variable Message Format (VMF) is one of those being investigated. An XML mapping to VMF is more difficult to develop because VMF is a "bit-oriented" messaging standard as opposed to the "character-oriented" structure of USMTF. Also most fields contained in USMTF messages are fixed in length whereas VMF messages have fields can be vary in length. A VMF can be only a few bits in length or can be several Mbits in length.

There are four alternative XML applications being considered for VMF [XMLV00]:

- Case 1: Develop a VMF to USMTF to XML-MTF Mapping.
- Case 2: Develop a VMF to XML-VMF mapping.
- Case 3: Develop generic XML Schema to support all VMF message definitions.
- Case 4: Use compressed XML-VMF document using COTS XML compression tools.

A decision will be made on the viability of XML-VMF mapping once these alternatives have been examined in detail.

7. XML Mapping Caveat

Significant problems arise with all of these XML to message format mapping initiatives. Focus must be placed on development and adherence to the standards defining the XML mapping (e.g., XML-MTF mapping) [HOP99]. If the standards are not strictly followed, the flexibility provided by XML can produce different system implementations of the XML mapping. The differences in implementation, no matter how slight would results in severe interoperability disruptions.

Detailed specifications must be developed and followed by all systems implementing the XML mapping. This is the only way full interoperability can be achieved between all systems using XML.

F. PREVIOUS DATABASE ANALYSES METHODS

1. Research Preparation

This thesis is part of a larger NPS research team supporting several related database analyses research topics as defined in the Joint Battle Management Initiative Assessment Plan Draft [JBMI00].

I teamed with Mr. Hamza Zobair to research the XML Schema Investigation topic as described in Section I.E. The objectives for this research were:

- Determine methods for assuring scalability of solution to legacy and migration C4I. This requires a method(s) that can support analysis and comparisons of both legacy and evolving common databases like JCDB.

- Determine what parts of a legacy system view could be materialized from previous shared schemas. This requires the identified method(s) to be able to identify common elements and physical schemas.

- Determine how to materialize those parts relevant to such an assessment. This requires the identified common data elements and schemas be integrated into a global common schema.

Our original research methodology was to jointly search for the required databases and any XML based analysis schemas we could find. Once found, these methods would be divided between the two of us and used to examine/compare

the identified databases. I was responsible for attaining and examining the AFATDS database and JCDB. Mr. Zobair was responsible for attaining and examining the MIDB and TDBM database.

2. Roadblocks Encountered During Research

The greatest challenge encountered during this research effort was trying to obtain a copy of the AFATDS database. It was quickly discovered that some of these legacy system databases are closely held products. After significant effort on my part, by other researchers on the team, and by NPS, the AFATDS database proved unattainable. Representatives from JBC also tried and failed to get a copy of the AFATDS database for this research effort. These efforts to get the AFATDS database spanned several months. The JCDB, however, was provided in multiple versions at the start of the research effort. Being able to attain only one of the two databases required to execute this analysis/comparison research efforts presented the first major roadblock encountered in this research effort.

The second major roadblock encountered in this research effort was hit while searching for XML based database analysis schemas that could be employed in analysis of the four specified databases. Our efforts focused on searching electronic technical libraries like ACM, IEEE, Society for Automotive Engineers, NPS's Dudley Knox Library, and Defense

Systems Management College Library for any related XML based database analysis schemas. The Tank-Automotive Research, Development, and Engineering Center's (TARDEC's) Technical Library was also employed to search out pertinent database analysis schema. The TARDEC Technical Library has (and used) automated search resources that searched several electronic technical libraries.

These searches did locate several technical papers describing database analysis techniques. Most were written in the early to mid 1980's. These papers focused on database analysis to support activities like data mining, data warehousing, and database integration. Most papers examined different aspects of the types of analysis methods that could be employed to support this research effort. The specific methods sought were ones that could distinguish and identify common data elements and physical schemas between heterogeneous C4I databases.

The located technical papers generally fell into three categories:

- Data Element/Data Hierarchical Searches: These papers provided the means to decompose the construct of a database schema, allowing the extraction of specific data elements and parent-child related data elements.

- Data Element Comparison: These papers describe methods of comparing multiple databases to locate common data elements in multiple databases.

- Database Integration: These papers usually examined methods to combine two databases into a single database.

Only one technical paper was found that described a method that would examine/compare the data elements and hierarchical relationships of two databases. Entitled "SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks"[SEM99], this paper examined most of the database analysis/comparison techniques sought in this research effort.

The problem with all of these papers is that none employed XML to support the database analysis and comparison. This was not a great surprise since most of the papers were written long before the 1996 inception of XML. But this led to the second major roadblock encountered in this research effort: How can an XML schema investigation of databases be conducted when no XML based analysis methods can be found?

3. An Opportunity for a New Analysis Method

Faced with these two major roadblocks, the objective for this thesis was refocused towards examining how XML could be employed in the development of an XML based database analysis and comparison method that would still meet the original objectives of this research effort. With the continuing development (and refinement) of XML and its associated XML based capabilities (i.e., XSL, DOM APIs,

InfoSet, etc.), the tools are available to define a new XML based C4I database analysis and comparison method. The remainder of this thesis will describe and demonstrate this new XML based C4I database analysis/comparison method. An additional objective for this method was to ensure it had broader application beyond C4I databases. This new method can be used to analyze and compare any XML document.

IV. XML BASED ANALYSIS AND COMPARISON METHOD DESCRIPTION

A. INTRODUCTION TO PROCESS

The original JBC/NPS research task was to determine methods to seek out XML methods for assuring scalability of XML solutions to legacy and migration to C4I database schemas [JBMI00]. As was briefly described in the previous section there are no XML based methods available to analyze database schemas. To ensure that the overall objective of this research effort was met a new, XML based database analysis and comparison method has been developed and will be described in this thesis.

1. Focus of Process

This process sought to demonstrate how XML can be exploited to analyze and compare common schemas between heterogeneous databases. It focused on the use of XML COTS software whenever possible to execute the analysis and comparison. This process also sought to reduce reliance on any legacy software.

The XML based analysis and comparison process description was divided into a sequential step by step process. Each step description began with a description of the components necessary to execute the process. These component descriptions focus on the individualized database

views to be analyzed along with the software, tools, and/or XML resources required to execute the analysis.

The process is described using the defined component and reinforced using examples whenever possible. The most detailed of these examples occur at the end of the process where views of the JCDB and MIDB are analyzed and compared using COTS based tools. The products of these analyses and the code developed to execute the analyses are included in the appendices of this thesis.

In some steps alternative analysis paths are available to the recommended analysis path. When this happens these alternative paths are briefly described, highlighting their benefits/ shortfalls and why they weren't included as part of the recommended process.

2. Database Analysis Aspects Not Covered

The JCDB and MIDB were not analyzed as part of this research effort. This is because the unique system specific database software was not available and the size of the two databases prevented detailed comparisons using existing computing resources. This is also because some the database information was unavailable. Instead this research effort focuses on smaller views of portions of the databases. These smaller views actually help facilitate the description of the analysis process.

Since required database software (i.e., Informix, Sybase, etc.) was unavailable or could not be used, examples supporting that particular step had to be built manually. When this was done it is identified as being a manual equivalent. The intent was to provide as detailed examples as possible and to support the description of the process.

Detailed description and analysis of the JCDB and MIDB are not provided as part of the research effort. This is because limited information was available on these databases. This was especially true for the MIDB. The portions of the MIDB extracted to build the MIDB View had to be built solely from the MIDB Data Dictionary. There were no entity relationship diagrams available to support definition of the database hierarchy. These relationships were drawn from the data dictionary.

Analyzing the entire JCDB would also be difficult due to its size. A JCDB View was built to simplify the analysis description. Also to strengthen the process description it was best to have a JCDB View equivalent in size to the one built for the MIDB.

B. DATABASE REVIEW STEP

This step will be where the databases to be analyzed are assembled and evaluated for completeness. The goal for this step is to have both databases roughly at the same level of detail. The term "level of detail" simply means

that the same type and detail of information is available for each database to be analyzed and compared. For example, if one database contains information describing the data type of the database attributes and the other database doesn't then it will be impossible to conduct a comparison of attribute data types between the two databases. It is best to balance the level of detail between databases whenever possible. This will ensure the analysis and comparison is equitably executed between both databases.

Balancing the level of detail requires close inspection of all documentation available on each database. Generally the available documentation is available in two forms.

First is the database's Data Dictionary that provides database schema and implementation information to support standardized database development and data usage [JDD00]. The data dictionary provides detailed information on the entity table and associated attribute and the associated relationship descriptions between them. For especially large databases, like JCDB and MIDB, using these relationship descriptions to identify relationships between more than a few entities is difficult.

To help alleviate this problem large databases also use Entity-Relationship Diagrams as the second form of database documentation. The diagrams provide graphical representations of the relationship hierarchy built into the database. For the purposes of this research these entity

relationship diagrams will focus on the logical representation of the data. This logical representation represents the inherent structure of the data, independent of the individual application [DOD98].

To balance the level of detail a visual inspection review and comparison between each of the data dictionaries and entity relationship diagrams must be conducted. If roughly the same information is available between the databases we can proceed to the next step in the process.

When the difference in the level of detail between the databases is significant it can jeopardize the analysis effort. There are two alternatives available if this should be the case:

One alternative would be to limit the subsequent analysis and comparison to only those portions of the databases that have the same level of detail. For example, if entity relationship information is not available for one database, the analysis and comparison can be limited executing an entity to entity comparison.

A second alternative would be to develop additional detail in the database lacking detail through searches for additional documentation or consulting with subject matter experts (SMEs). This additional data, if discovered, may have to be manipulated into a format that is comparable to the other database.

1. JCDB/MIDB Comparison

In this research effort the JCDB and MIDB were chosen for comparison. Smaller portions of the two databases were chosen to execute this analysis and comparison effort. This was done for the following reasons:

Database Size: The extremely large size of the two databases would add unneeded complexity and confusion to the description of this analysis and comparison process. It was best to focus on portions of the databases that can best be used to describe the process.

Limited Access to Data: Specific software called "ERwin" was required to view the JCDB logical representations. This researcher was only able to gain use of a limited two-week trial version of the ERwin viewing software. This trial version of the software limited the amount of entity relationship detail that could be extracted from the JCDB logical representation.

MIDB Entity Relationship Diagram: The entity relationship diagrams for the MIDB could not be located for this research effort. The logical representations of the smaller portions of the database built had to be constructed from the MIDB Data Dictionary relationship descriptions. It would have been impossible to build the entity relationship diagrams for the entire MIDB. For the purposes of this research effort I was required to make certain assumptions

in the development of these entity relationship diagrams that could result in slight deviations from the actual relationships contained in the MIDB.

To develop these entity relationships diagrams the DoD 8320.1-M-1 Data Standardization Procedures and the Integration Definition for Information Modeling (IDEF1X) was used as a means to standardize the logical representation of the database entities.

DoD8320 provides the procedures for developing, approving, implementing, and maintaining DoD data standards. These data standards provide the framework for how the data will be formatted for implementation within the information system [DOD98]. The IDEF1X defines how to produce a graphical information model that represents the structure and semantics of information within an environment or system [IDEF93].

2. JCDB View

To build the JCDB View, portions of the JCDB were extracted from the data dictionary based on the limited entity relationship diagrams available. The entity relationship diagrams were reviewed and modified slightly to better conform to DoD 8320 and IDEF1X. Examples of the extracted portions from the JCDB Data Dictionary and associated entity relationship diagram can be found in Appendix A.

3. MIDB View

After extensive review of the MIDB Data Dictionary, portions were extracted that were similar in nature to the extracted portions of the JCDB. These portions focused on the Target Assessment and Battle Damage Assessment Report (BDAR).

Since no MIDB entity relationship diagrams were available, one had to be developed from the relationships described in the data dictionary. Appendix B contains examples of the extracted portions of the data dictionary and the developed entity relationship diagram.

4. Conclusion

With the completion of this step there are now two database views that contain approximately the same "level of detail". As describes earlier, this will ensure subsequent analyses and comparisons are based on comparable data.

C. DATABASE CONVERSION TO XML

The two or more databases with roughly the same level of detail, developed in the last step must now be converted into XML documents. This is relatively simple process if the database software (i.e., Informix, Sybase, etc.) retains an integrated XML translation capability. As described in Section D, many of the database software manufacturers are integrating XML translators in one form or another to support internet applications of their databases. For the

purposes of this research, the conversion to XML provides the common basis upon which the analysis and comparison can be executed. This conversion also supports one of the primary objectives of this research effort to bring XML into the analysis and comparison process.

1. Database to XML Translation

If the database software in question has an XML translation capability, then the XML translation of the database to be analyzed requires the analyst to execute the translation process as specified by the database software. Before converting the databases, each database to XML translators should be reviewed to confirm they conform to the same XML Recommendation. Currently, only XML Recommendation 1.0 has been released. In the future, as XML Recommendation 1.0 is updated, there may be situations where older database software products maintain translators built to outdated recommendations while newer database XML translators are built to the latest recommendation.

It is critical that automated methods, like the built-in XML translators, be used to translate databases like JCDB and MIDB. The size and complexity of these databases would make any manual XML conversion impossible. Additionally, the XML document equivalent of the database would be even larger. In Joint Battle Management Initiative (JBMI) experiments conducted in July 2000 it was estimated that the

XML equivalent growth from a USMTF message was approximately 10 to 1. While USMTF is not a database, it still has to maintain specific data type and structure to convey a specific message. Likewise a database maintains specific entity and attributes in a specific structure. Additionally, the database must maintain the relationships between these entities to convey the hierarchical parent/child relationships. It can be hypothesized that the XML growth for relational databases like JCDB and MIDB would be even greater than 10 to 1. This was one of the reasons smaller views of the databases were extracted for use in this research project.

Once translated, portions of each XML document should be manually examined to ensure the translations were executed as expected. This manual inspection would require cross-checking between the database data dictionary, the entity relationship diagrams, and the XML document to ensure the entities, attributes, and hierarchical relationships are captured in the XML documents. Only one or two portions of the database and XML document needs to be examined to ensure the translation was successfully completed. This step is completed once all the databases have been converted into XML documents and they have been successfully reviewed.

2. XML Translation Alternative

An alternative automated translation process is required if the particular database software product does not have an XML translation capability or when the manual inspection of the translated XML document resulted in unacceptable documents.

This alternative focuses on utilizing Microsoft's Open Database Connectivity (ODBC) APIs built into most database software products. The ODBC APIs, based on Structured Query Language (SQL), provides call functions that can be used by database software products to manipulate the particular database [ODBCWP] into an ODBC structure. Then a shareware product called ODBC2XML can be used to convert an ODBC formatted database to XML. Being that ODBC2XML is shareware there is no assurance that the converted XML document fully represents the database. A thorough examination of XML documents must be made to ensure its validity. For the purposes of this research this conversion alternative will not be examined further.

3. JCDB and MIDB to XML Translations

Without access to any of the necessary database software products (or the specific databases) it was impossible to execute either of these translation alternatives. This does not impact this research since as described earlier, this step in the process is an automated

process using resources already built into the database software product.

As an alternative the XML documents to be used in this research project were manually developed from the JCDB and MIDB views included in Appendix A and B. Due to the lack of data on the JCDB and MIDB, certain assumptions had to be made when developing these XML documents. These assumptions centered on defining certain hierarchical structures of the XML document and the data contained in the XML documents. These assumptions do not impact objectives of the analysis and comparison process, which is to demonstrate how common XML based schema can be extracted. Another assumption made while developing these XML documents was to exclude the development of DTDs or XML Schemas. Including or omitting the DTDs and Schemas do not impact this analysis and comparison process. In the case of this research effort they only add additional complexity that might distract from the process description.

4. Conclusion

This step is completed with the development of multiple well-formed and comparable XML documents. The JCDB and MIDB XML documents are included in Appendix C and Appendix D. These documents provide a common basis upon which analysis and comparisons of two (or more) databases can be executed.

D. ENTITY/ATTRIBUTE ANALYSIS AND COMPARISON

As described in Section I, the original objective of this project was the execution of an XML based analysis and comparison of the AFATDS, GCCS TDBM, GCCS I3 (MIDB), and the JCDB. I teamed with Mr. Hamza Zobair to accomplish this task. Due to difficulties described Section III.F.2 we each decided to broaden our research topic to define our own analysis processes of the JCDB and MIDB. Mr. Zobair chose to research a process to analyze and compare database entities and attributes. I chose to research a process to analyze and compare the hierarchical relationships of the databases. Both analysis methods are required when comparing and locating common portions of databases.

1. Introduction

This step in the overall XML-based analysis and comparison process was researched and described by Mr. Hamza Zobair in his thesis entitled: "An Approach for Matching Corresponding Attributes in Legacy Heterogeneous DoD Databases" [HZ01]. This process to conduct an entity and/or attribute analysis and comparison will only be briefly described in this thesis. I highly recommend reading Mr. Zobair's well-written thesis for the full description of this analysis process.

2. Process Description

This step in the overall XML based analysis can be conducted prior to or in parallel with the previous steps described so far. The basis for comparing the entities and/or attributes of databases is the entity and attribute tables located in the database data dictionary. The first step in the process requires the restructuring of the tables into comparable structures. Depending on the amount of restructuring and size of the tables, this can be a very time consuming process. It is recommended that automated methods be used whenever possible.

Once restructured, the tables are compared using automated comparison tools. These tools would search and compare the tables and identify potential matches of common entities and/or attributes. These tools employ user developed thesauruses and data clusters as the basis to execute the entity and attribute comparisons.

Once potential matches are identified, the matching criteria are refined based on a manual review of the matches. A final analysis is then conducted to identify the best possible matches. These matches are then reviewed by domain experts to validate or discount the matches.

3. Conclusion

Mr. Zobair's research examined the attribute tables of the JCDB and MIDB. These attribute tables are significantly

more complex than the entity tables. The same analysis process would be used to analyze and identify common entities.

These identified common entity and attributes become the "search keys" critical to the remainder of the database analysis and comparison. The use of these search keys will significantly simplify the search of the extremely large XML documents built from the databases.

Finally, it must be stressed that to fully understand this process it is important that Mr. Zobair's thesis be reviewed.

E. HIERARCHICAL EXAMINATION

At this point in the process it is important to review what has been developed so far:

- Establishment of two or more databases that have been reviewed and determined to be roughly comparable.
- Development of XML documents built from the comparable databases. If built from the JCDB and MIDB, these documents would be very large.
- A list of search key developed through the entity and attribute analysis.

What is required now is an automated process to search these large XML documents to locate desired portions of the database based on given search keys. Once the desired entities and or attributes are located this process must

then be able to extract and present them to an SME that has a detailed understanding of that portion of the databases.

This identification and extraction process must be able to maintain the hierarchical composition of the XML document. This is the overall objective of the research effort, identification, extraction, and comparison of specific hierarchical relationships of multiple heterogeneous databases.

One of the reasons for converting the databases to XML documents is that there are XML based tools/resources available to examine and manipulate the hierarchical composition of XML documents. The key XML based resource to be used in this research effort is called the Document Object Model (DOM).

1. What is a DOM?

From the W3C DOM web page, the DOM is defined as "...a platform and language - neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be processed and the results of that processing can be incorporated back into the present page." [W3CWP] More simply put, the DOM is a specification that defines how an XML (or HTML) document can be parsed into a node tree representation of that document and analyzed.

The node tree representation of the document begins with the root element of the XML document set as the root of the tree. The children of the root branch out to nodes. They are called "nodes" because each of the XML document components is parsed into their own individual node. Each node object implements a node interface. The following figure shows all the node interfaces that can be used to build a DOM node interface tree [XMDB00].

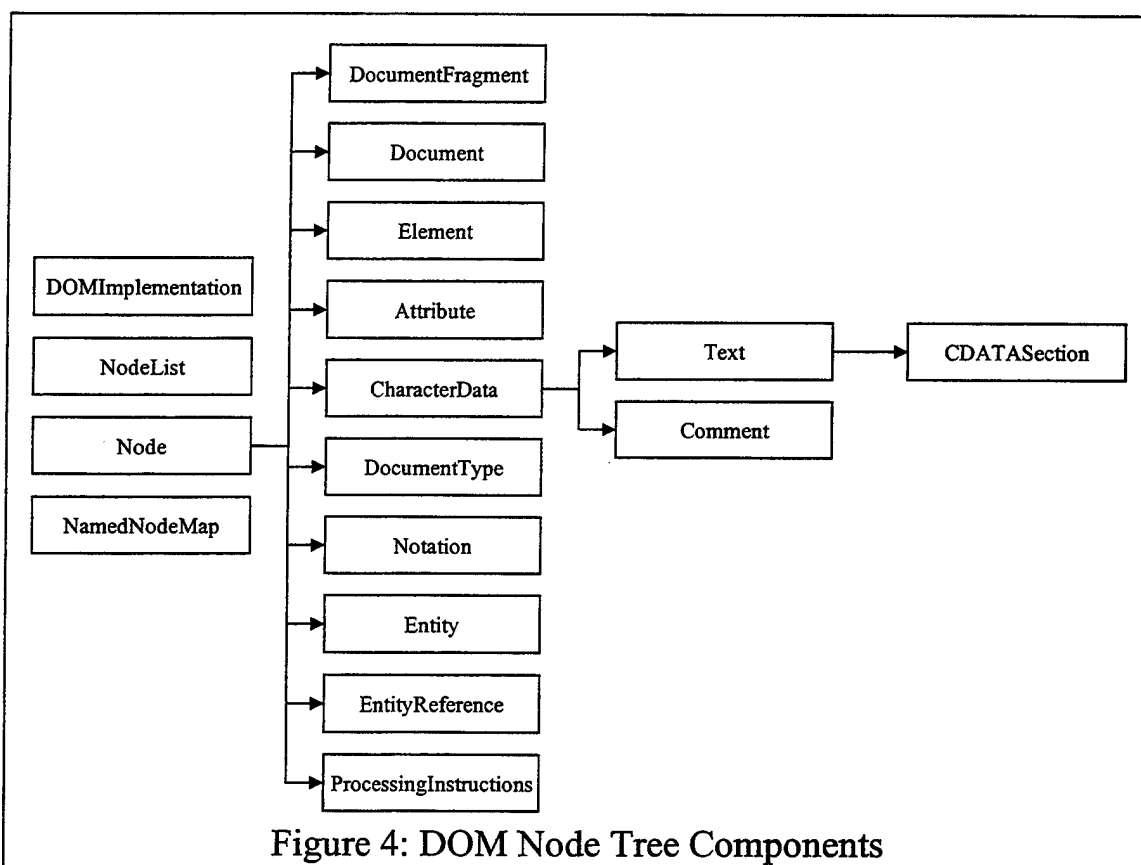


Figure 4: DOM Node Tree Components

These node interfaces provide the points where the DOM node tree can be navigated and manipulated. Scripting languages, like JavaScript, can be used to invoke DOM APIs

that provide the means to move through the tree and modify it. An example how this works will be described later.

a) DOM Recommendations

The DOM Recommendations are managed by the W3C. The W3C is a consortium of over 500 members from 34 countries that produce standards setting, interoperable technologies through consensus. Their membership is comprised of industry, government, citizens groups, and other organizations committed to the development of the Web [W3CWP]. The objective document the W3C publishes is the "recommendation" as the defining and locked document that describes a specific web based technology (e.g., DOM). When developing the recommendation, a W3C technical working group made up of experts in that technical field, posts the working documents called specifications to the web site. Anyone is welcomed to submit comments on the specifications. Through consensus the working group determines what specification modifications are required based on their own individual developments and submitted comments. Once a specification is finalized and approved by the working group it is posted as a "recommendation". The DOM Recommendations provide the interface definitions for the DOM API libraries. The W3C has posted three levels of DOM Recommendations and Specifications.

The DOM Level 1 Recommendation, issued on 1 Oct 1998, defines the foundation set of interfaces to navigate and manipulate the XML (and HTML) documents. A second edition of the DOM Level 1 specification is now under development and is posted on the W3C Web Site for review.

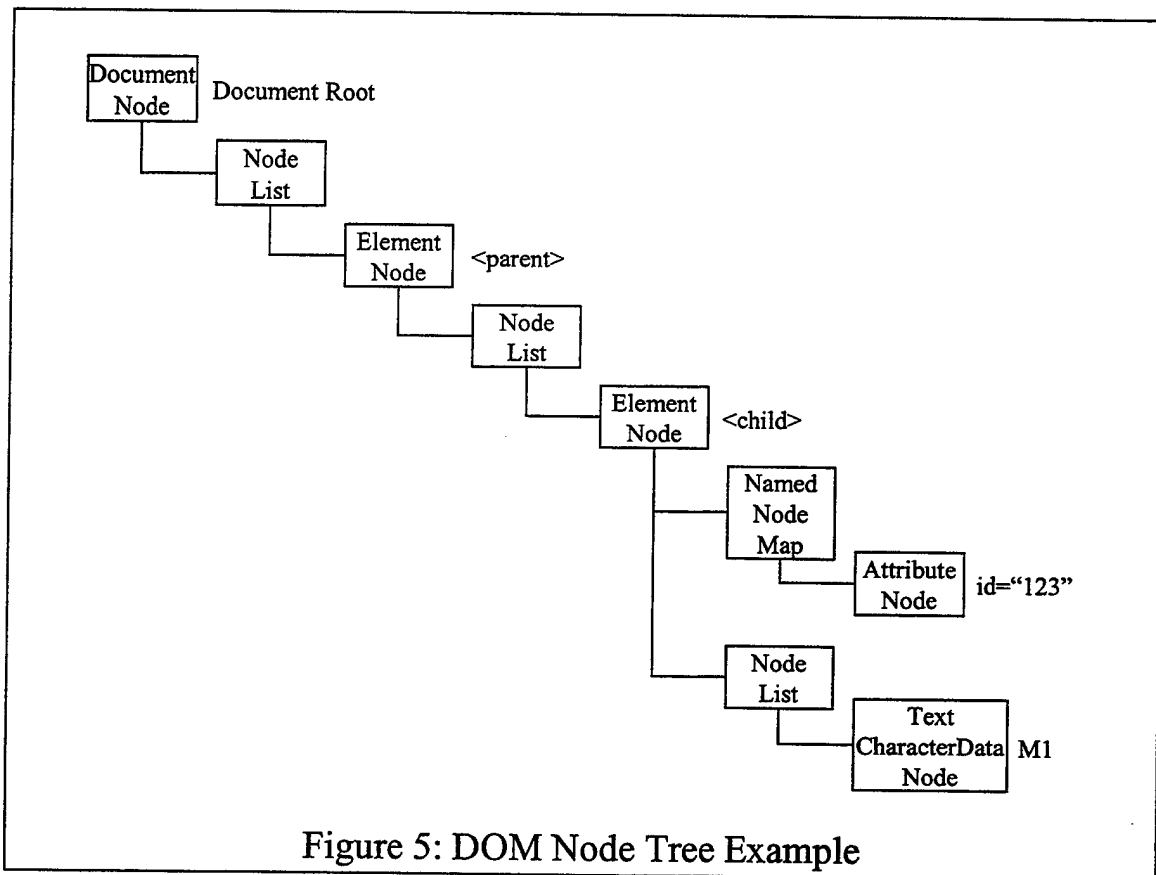
The DOM Level 2 Recommendation builds upon the DOM Level 1 Recommendation by defining additional interface definitions. It includes a style sheet, object model, and defines functionality for manipulating the style information attached to a document. It also provides support for XML Namespaces [RCWP]. The Level 2 Recommendation is comprised of the Core View, Style, Event, Traversal-Range Recommendations all issued 13 November 2000.

The DOM Level 3 Specification (not a recommendation yet) will define loading and saving interfaces and content models with validation support. Also to be addressed will be document views, formatting, key events, and event groups. The Level 3 Specification Working Drafts posted are the Core (posted 1 September 2000); Content Models and Load and Save Interfaces (posted 9 February 2001); and Views and Formatting (posted 15 November 2000).

b) DOM Example

An example of a DOM Node Tree can be built from the following XML document (adapted from example in [XMDB00]):

```
<Vehicle>
  <Tank id="123">M1</Tank>
</Vehicle>
```



The nodes in Figure 5 present the objects and interfaces where the DOM can be examined and manipulated. Each box is considered a node object. The names in the boxes are the interfaces that will be implemented by the

objects. The NodeList object controls a list of nodes below it. This NodeList will change as nodes are added or deleted. The NodeNamedMap controls unordered sets of nodes referenced by their attribute names. The NodeNamedMap also changes based on the addition and deletion of nodes.

c) Examples of Interfaces

The following are examples of the interfaces related to various node objects. The most fundamental object in the DOM is, of course, the Node. The node retains certain properties and methods that will allow the traversal of the tree, obtaining specific information on the node, and manipulating the node. The following are a few of the node properties (adapted from [XMDB00]):

Property	Description
nodeName	Returns value of specified node.
nodeType	Returns type of specified node.
childNodes	Returns the node list of specified node. If no children, returns empty node list.
previousSibling	Node immediately before current node.
nextSibling	Node following current node.

Using the properties listed in the table the following: "previousSibling.nodeName" would return the name of the previous sibling's name. The employing of the

properties "nextSibling.nodeValue" would return the value of the following sibling node.

These two examples provide demonstrations of how the DOM can be navigated. The properties "previousSibling", "nextSibling", "parentNode", "firstChild", and "lastChild" are the primary means used to navigate through the DOM node tree.

Besides the properties, the node also has methods that can be used to manipulate the DOM. The following table provides some examples:

Method	Action
insertBefore(newChild, refChild)	Inserts new child before current reference child.
replaceChild(newChild, oldChild)	Replaces old child with new child. Returns the old child.
cloneNode(deep)	Returns a duplicate of node. Deep is a boolean value. If false, returns node. If true, the node and entire subtree under the node is returned.

The properties and methods listed are only a few of those available in the DOM Recommendations and Specifications.

d) Products Employing DOM APIs

With the DOM Recommendation (DOM Level 1) only being available for just over two years only a few commercially available DOM tools and applications have been developed. Probably the most basic and most universally available is Microsoft's Internet Explorer 5.0, which has integrated an XML parser and DOM APIs (Note: Netscape also has integrated a limited XML parser and DOM APIs). The reason Microsoft has been able to integrate XML parsers and DOM APIs before other manufacturers is because they began their integration efforts long before the actual recommendations had been approved. In some cases, they risked building in XML and DOM capabilities based only on requirement documents.

Besides Internet Explorer, there are other DOM tools available. A few of these DOM products were listed on the web. Even though most of these DOM products have unique platform/software requirements that prevented detailed investigations in this research effort they do demonstrate that DOM APIs are being widely adopted for use in the XML community. As the other DOM Recommendations are released the number of DOM products/application will surely grow.

e) Problems With DOM

The primary problem with DOM APIs is that when a DOM node tree is created it can be 5-10 times the size of

the originating XML document. Earlier I hypothesized that an XML document built from a database like JCDB could easily be 10 times larger [XMDB00]. Combine the growth from database to XML document to DOM node tree and the final DOM tree could be 100 times the size of the original database.

Another problem with using DOM APIs is that they are still evolving. There are still several specifications defining new APIs undergoing revision. Additionally, the DOM Level 1 Recommendation is already undergoing revision in its second edition. Potential developers desiring to integrate DOM APIs may continue to wait until all the DOM APIs become more stable.

f) DOM Problem Solutions

Memory Usage: The DOM memory usage will become less of a problem as the computing technologies continue to grow. Personal computers with 1.5 Ghz processors, 300 MB RAM, 100 GB hard disks, and 500 GB DVD read/write drives can be purchased today. These computing capacities can be expected to double each year for the next several years. With this level of computing power/capacities available to anyone, the size of the DOM should not be a problem.

Unstable Recommendations: Recognizing the potential of DOM APIs, large software developers, like Microsoft and Netscape, have already integrated DOM capabilities into their browsers. This was even before any

recommendations had been approved. As XML grows in acceptance, so will the use of DOM. Software developers will have to commit to incorporating DOM APIs into their products if they want to take full advantage of the XML.

Alternative to Using DOM: Another method available to search and manipulate XML documents is to use the Simple API for XML (SAX). The SAX is an event-based interface that serially processes XML documents and notifies the application calling the SAX when a certain event has occurred. The DOM on the other hand loads the entire document into memory and manipulates it. The SAX's serial processing approach eliminates the memory burden associated with the DOM. It also allows the SAX to process an XML document of any size. Another benefit to using SAX is that it provides several APIs to navigate and manipulate an XML document.

The use of SAX does have shortcomings. First is that SAX is not associated with any standards and/or consortium bodies like the W3C. As a result, the SAX has no design stability since the SAX can be changed at any time. Another problem with using SAX is that complex searches of XML documents are difficult. Since the SAX process the XML document serially, multiple searches might have to be made to find a single element. For example, suppose a parent element of a child element must be found. The SAX would have to process the XML document to find the child element

and then process it again to find the parent element. This will increase the SAX's overall processing time of an XML document.

g) Selection of DOM API

During this research effort both the DOM and SAX had to be examined to determine which would best serve this analysis and comparison process. The DOM was chosen because it presented more capability to navigate and manipulate an XML document. Additionally, the DOM's memory usage problem would not impact this research effort since smaller views of the JCDB and MIDB were being used.

2. Process Description Introduction

The objective of this step in the process is to employ DOM APIs to examine and manipulate the XML documents built from the database views. This can be accomplished with the development of scripting code to invoke the DOM APIs. This was accomplished using relatively few lines of code. The majority of the code was necessary to account for the output and storage functions that aren't yet available because the associated DOM Recommendations have just been approved or are nearing approval. Once all the DOM recommendations are incorporated into a software application, it will be a simple task to streamline the code to make it much more efficient.

Microsoft's Internet Explorer with its built-in XML parser and DOM APIs were chosen as the software application to be used in this process. At this time there are no other commercially equivalent XML parsers available for use on a standard personal computer using Windows '95. Internet Explorer 4.0 and Netscape 6.0 only have limited XML capabilities. There are some shareware XML parsers available on the web, but as with most shareware products their functionality and reliability is questionable since there is no commercial or technical rationale for the developer to maintain the product.

JavaScript was used as the scripting code to enable the Microsoft XML Parser (MSXML) and invoked the DOM APIs. The JavaScript was used to develop the code necessary to import the particular database XML documents into Internet Explorer 5.0, to parse that XML document into a DOM node tree, to analyze that DOM node tree for a given search key, and to output results of that search. The script code developed for the analysis process used in this research was adapted from code found in the book "XML IE5" [XMLIE99]. The code was extensively modified to support this research effort's need to execute an efficient search and manipulation of the XML documents. The code used to output the located portions of the XML document is relatively unchanged from the code in the book. It provides the capability to import the products of the analysis into this thesis.

3. Process Description Synopsis

To facilitate comprehension of the process, each step in the process will be examined in detail by stepping through the JavaScript code developed to execute the analysis. The following is a quick overview of the process:

1. Parsing the XML Document: The XML document is parsed into a DOM node tree.
2. Node Tree Search: Using DOM APIs the node tree is searched for desired elements. A node list is developed that contains all nodes that match the desired search key.
3. XML Fragment Build: An XML fragment is built by deep cloning the individually located nodes. This cloning produces copies of the located node and all of its children. The cloned node is attached to the fragment. The fragment is complete when all located nodes (and children) have been attached.
4. Fragment Decomposition: The analysis process concludes when the built XML fragment is broken down into the individual nodes, converted to text outputs, and displayed.

4. Analysis Process Description

This process description will examine key portions of the JavaScript and DOM APIs invoked to examine the XML document. The specific functions of the DOM APIs will be emphasized where it facilitates the search and manipulation of the document. The HTML code used as part of this analysis will be described when it impacts the involving of the DOM APIs.

The complete code with extensive comments can be found in Appendix E. Through the comment lines the code has been divided into distinct sections. More critical sections like the ones involving the DOM APIs will be described in detail. Other sections that are only required to support the execution of the code (e.g., the HTML code) will be briefly described.

a) *XML Document Import*

This section contains the initial portion of the HTML Head. As discussed earlier, HTML is used to support the JavaScript execution of the XML parser and DOM APIs.

```
<XML ID="domSearchList" SRC="YYYY.xml"></XML>
```

This line informs IE5.0 to invoke its built-in XML parser. The IE5.0 used in this research project is an early version of the XML parser. If a later version of IE is used to execute this code, this line may have to be changed to invoke the later version of XML parser (i.e., MSXML2, MSXML3, etc.) Examples of how to invoke these later versions of MSXML can be found in Professional XML Databases [XMDB00].

This same line also instructs IE5.0 what XML file to import. Each time a different XML document needs to be examined the "YYYY" will have to be changed to the name of that XML document. The best way to modify this code is to use the Notepad Application found on most personal computer

platforms. When the HTML code is modified, be sure to save it as a text file with an HTML extension. The XML document that is to be parsed must be located in the same folder as the HTML file.

The last line of this section instructs IE5.0 that JavaScript will follow.

b) XML Parsing

This section supports the parsing of the XML document into the DOM node tree and to raise any parsing error conditions if the parsing was unsuccessful.

```
objXMLData = document.all['domSearchList'];
```

This line executes the parsing of the XML document. The remainder of the code checks for and outputs any parsing errors. The "parseError" API is an extension built specifically by Microsoft to support Internet Explorer. It is not part of the W3C DOM Recommendations. It was included because it was simple to import as is from the original code and proved beneficial when parsing the XML documents. It identified several format/structure errors in the XML documents. This was the only Microsoft IE specific DOM APIs employed in this code. All other DOM APIs used are included as part of the W3C DOM Level Recommendation 1.0.

c) HTML Search Function Call

The single line contained in this section calls the "searchDocument" function in the following section. It also returns the assembled strNodes variable that contains the parsed XML fragment that will be described in a later section.

The searchDocument function sends two sets of data to the function. First is the DOM node tree to be examined. The second is the search key to be matched as the DOM node tree is searched. The search key should be the same as was discovered in the Entity/Attribute Analysis and Comparison Section. The search key will have to be changed every time a different element needs to be located in the DOM node tree. It's best to the Notepad Application to change the search key. This search function is case sensitive so the search key must be input exactly as was found in the Entity/Attribute Analysis and Comparison. Also, the quotation marks must be used with the search key. The following example is taken from the code in Appendix E:

```
divResults.innerHTML = searchDocument(objXMLData, "TARGET-  
ENGAGEMENT-ASSESS");
```

d) DOM Tree Search

This section, along with the following four sections describes the searchDocument function. This

function is the primary function developed for this research project. It executes the search for the common elements, extraction of the common elements, and the calling of the functions required to build the output of those extracted elements.

This section establishes all the variables used in this function. The key variable "objFrag" is created as an XML fragment. It is considered a fragment since it is not a well formed/valid XML document. It will contain only a root element and added elements. This fragment is a holder of the common elements located during the DOM node tree search.

```
listNodes = theRoot.getElementsByTagName(searchKey);
```

The line above calls the DOM API to search the DOM node tree for all the elements that match the "searchKey". It stores the located common elements in a node list. For example, if the DOM node tree contained 4 separate occurrences of the element <CAR>, the `getElementsByTagName("CAR")` would return a node list containing those four specific <CAR> nodes.

The use of the `getElementsByTagName` API will be highly beneficial when searching extremely large DOM node trees like those that would be built from JCDB and MIDB. The `getElementsByTagName` is only called once during the entire analysis process. Only having to search the DOM node tree once makes this developed search process very

efficient. The resulting node list used from this point on contains the specific information of the located elements (e.g., location in the tree).

e) Notification of Found Elements

This section examines the `listNodes` variable containing the built node list to see if it contains any matched nodes. The "`listNodes.length`" returns the number of nodes in `listNodes`. If the `listNodes.length` is greater than 0, an alert window is opened displaying how many common elements were found. If `listNodes.length` equals 0, then two alert windows are opened providing additional guidance. Even if the `listNodes.length` is equal to 0, the code continues to execute until complete. Again this does not impact execution time since all subsequent executions triggers loops that use the `listNodes.length` as the upper limit of the loop. So when `listNodes.length` is equal to 0, the loops do not execute.

f) Extracting Found Nodes/Building XML Fragment

This executes a loop to extract the individual nodes from the node list and adds them to the previously created XML fragment. The nodes listed in `listNodes` are numbered starting with zero. So `listNodes(0)` identifies the first located node, `listNodes(1)` identifies the second and so on.

Each time the loop executes for I=0 through I=listNodes.length the following line is used to take the found node and clone it:

```
objNode = foundNode.cloneNode(true);
```

As described earlier, the cloneNode(true) API makes an exact duplicate of the node and of all the descendant nodes. For the purposes of this research effort, it is critical to extract these descendants since they will be an important part of the database to database comparison.

```
objFrag.appendChild(objNode);
```

The line above attaches the cloned node (and its descendants) to the XML fragment.

The loop continues to execute until all the nodes contained in listNodes have been cloned and added to the XML fragment. An alert window is opened to display the entire XML fragment. The person conducting the analysis can quickly scan the alert window to determine if the desired information was found in the fragment. If the alert window is large the "OK" button may be off the screen. If this happens simply hit "ENTER" key to close the window.

g) Building the Output

This section executes the same loop as before to again extract the nodes in listNodes. Each execution of the loop calls the "showChildNodes" Function sending the node from listNodes. This function was modified from code found

in the XML IE5 book [XMLIE99]. The showChildNodes Function returns a text output containing the parsed details of the node sent during the function call. This text output is appended to the strNodes variable.

The searchDocument Function is completed when the listNodes loop has finished. The strNodes variable containing the assembled text output of the all the located nodes is returned for display.

h) Parsing Node for Output

This section contains the definition of variables and the assembly of the information on the nodes sent to the showChildNodes Function. The strNodes variable is continuously appended with information on the node. The following information is appended to strNodes:

API or Function Call	Action
getIndent(intLevel)	A function call to improve readability of output. Will be describe in more detail in getIndent Function Section.
objNode.nodeName	Returns name of objNode.
getNodeTypes(objNode. nodeType)	objNode.nodeType returns a number between 1 and 12. These values are predefined values that are associated with the individual node types that

	can be found in a DOM Node Tree (see figure 4). The getNode type function is called sending node type integer. See the getNode type Function description.
objNode.nodeValue	Returns a text equivalent of the value contained in the node. If no value is found, null will be returned.

i) Output Attribute Node Information

This section returns any attribute information related to this node. Nothing will be added to strNodes if there are no attributes associated with this node.

Invoking the objNode.attributes API develops an attribute list containing all the attributes associated with the objNode. This list, called the NamedNodeMap, functions just as the node list.

The objAttrList (the attribute list) is first checked to see if it contains any attributes. If objAttrList is not equal to "null" the function continues to parse the attributes. A loop is then called to parse all of the attributes contained in the attribute list. The same DOM APIs used to examine the nodes in the node list are used to examine the attributes in the attribute list. Each loop execution appends the attribute information to the strNodes

variable. Review of the JavaScript code in Appendix E is recommended to see the similarities.

j) *showChildNodes Function Called Again*

When the nodes were originally cloned, their descendant nodes were also cloned. This section examines the node to determine if it contains any children nodes. If so, the `showChildNodes` function is called again to get information on that child node. If that child node contains its own children nodes the `showChildNodes` function will be called again to get their information. This is what is called a "recursive" function call. This basically means that a function calls itself. In the case of this analysis, the recursive function calls will continue until all the descendant nodes of the original node have been located and their parsed node information has been appended to the `strNodes` variable.

k) *getNodeType Function*

The `getNodeType` Function returns a text output describing the type node being examined. The specific output is based on the integer identified by the `objNode.nodeType` and `objAttrList(intAttr).nodeType` DOM API calls.

1) *getIndent Function*

This section contains the *getIndent* Function that is used to insert indents into the *strNodes* variable to improve readability of the output. Each time the *showChild* Function is called, the indentation is increased. This makes it easier to distinguish the parent nodes from the children nodes.

This section also completes the JavaScript used to execute the analysis of the XML document. The remainder of the sections describe the remaining HTML code required to execute the analysis.

m) *parseXML Function Call*

This section calls the *parseXML* Function described previously. It also calls the *searchDocument* Function and displays the completed *strNodes* variable.

n) *XML Document Button*

This HTML code displays a button that when selected displays a separate page containing the XML document that was analyzed. This code was left in because it helped during code debugging. It is strongly recommended that this button function be disabled as described in the comment line of the code when analyzing extremely large XML documents, like those built from JCDB and MIDB.

5. Summary of Analysis Process

The code just described provides a standardized search and manipulation process that can be used on any XML document. This code provides several benefits:

One benefit of using a standardized analysis process is that the outputs are standardized. This simplifies the process of comparing the located nodes from heterogeneous databases.

This analysis process only searches the DOM node tree once. The node list built during the initial search is used from that point on to extract the located nodes. This provides a timesavings when searching extremely large DOM node trees built from databases like JCDB and MIDB are analyzed.

This process extracts small, comparable outputs from large databases. This eases the efforts to compare the potentially common elements.

F. JCDB AND MIDB VIEW ANALYSIS

1. Analysis Component Review

The following describes all the components developed to support this analysis:

Entity Relationship Diagrams: These diagrams were developed for the selected views of the JCDB and MIDB. These particular views were chosen for their inherent commonality that would help facilitate the description of

this process. These views focused on the Target Engagement and Target Engagement Assessments domains.

The particular view of the JCDB Entity Relationship Diagram was extracted from the provided JCDB Entity Relationship Diagram. The MIDB Entity Relationship Diagram was unavailable for this research effort. As a result the entity relationship diagram of the selected MIDB View was built manually from the relationships identified in the MIDB Data Dictionary.

XML Documents: The XML documents for the JCDB and MIDB Views had to be built manually because the databases were not available for this research effort. As described in Section IV.C, different automated methods are available and should be used whenever possible when trying to convert extremely large databases. This is another reason specific views of the databases were chosen for the research effort. These smaller views supported the manual development of the example.

The XML documents were developed in a manner that supported description of the analysis process. Also these XML documents were developed without DTDs or Schemas. The developed analysis process does not require the DTDs or Schemas. For those who are concerned about the lack of DTDs or schemas, this analysis process does not preclude the use of the DTDs or Schemas.

Search Keys: As described earlier, the search keys would be provided as products of the Entity/Attribute Analysis and Comparison process described in Mr. Hamza Zobair's Thesis: "An Approach for Matching Corresponding Attributes in Legacy Heterogeneous DoD Databases". Due to the concurrency of Hamza Zobair's research and mine, his described analysis process was not used as part of this research effort. Instead search keys were manually selected based on their inherent similarities. The search key selected for the JCDB View was "TARGET ASSESSMENT". The search key selected for the MIDB View was "TGT_DTL_ASSESS".

2. JCDB Analysis

The first step in the JCDB analysis is to insert the name of the XML document to be analyzed and the given search key. Using Internet Explorer, open the HTML file containing the analysis code. To execute the analyses simply click the "GO" button at the right side of the address bar.

The first alert window to open displays how many "TARGET ASSESSMENT" nodes were located in the JCDB DOM node

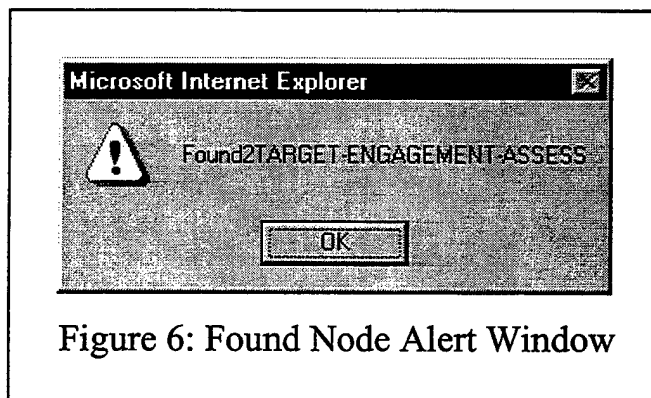


Figure 6: Found Node Alert Window

tree. Figure 6 shows how many TARGET ASSESSMENT were nodes located in the JCDB Node Tree.

Click the "OK" button or hit the "Enter" key and the second alert window opens displaying the located nodes and all their descendant nodes. The Figure 7 shows that some of

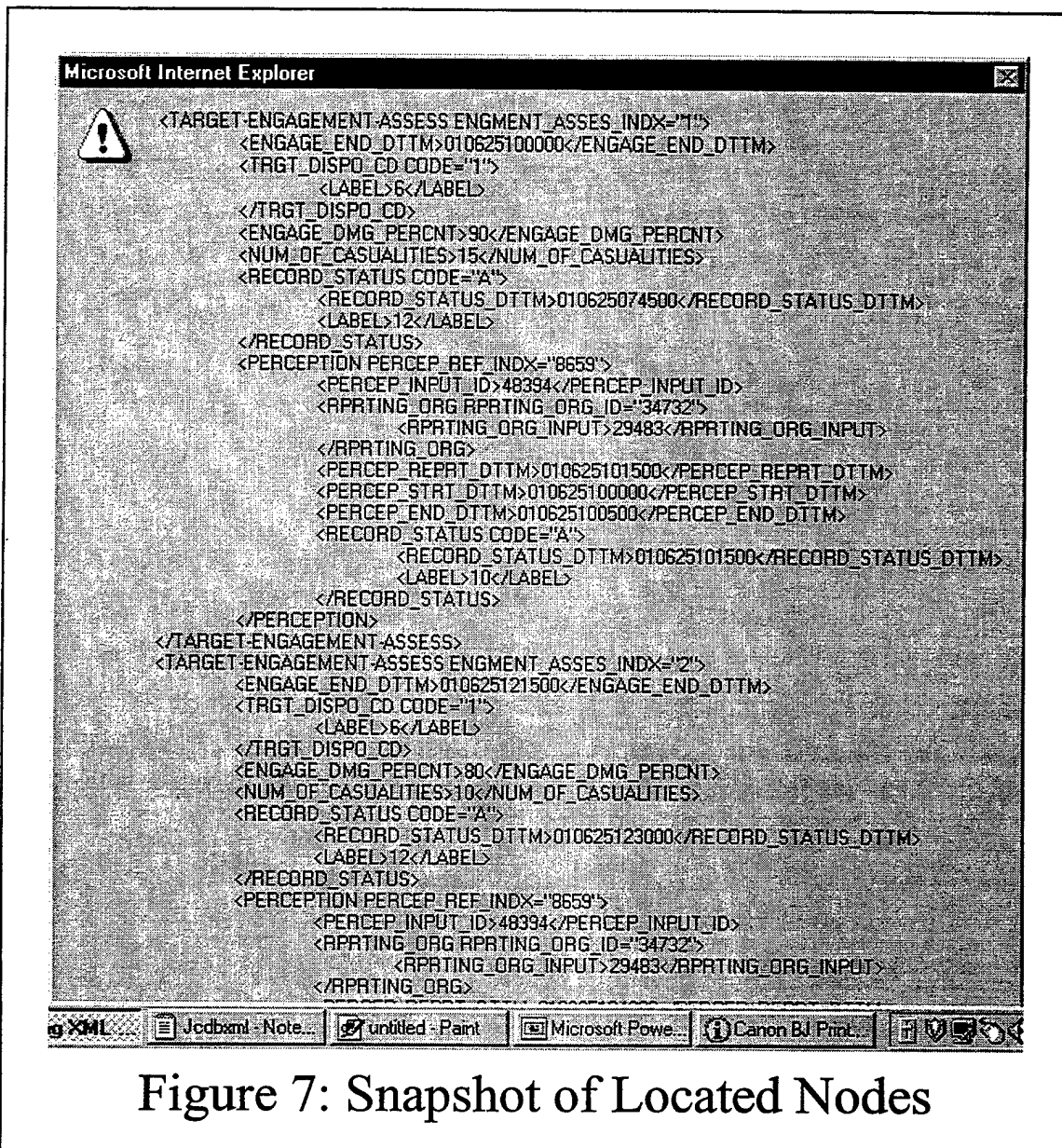


Figure 7: Snapshot of Located Nodes

located information can be quickly reviewed to determine if its the type of information desired.

Note that some of the information is located off the screen. This is not important since a more thorough output will follow. Since the "OK" button is located off the bottom of the screen the "Enter" key will have to be hit to close the window.

The nodes displayed in the previous alert window are then parsed into the final standardized output and displayed in the Internet Explorer window. This output can be printed or copied into other software products for the following comparison. Due to its size, the output from the JCDB analysis is included in Appendix F. This completes the process of searching the JCDB for specified nodes.

3. MIDB Analysis

The exact same analysis process is used to analyze the MIDB View. The only variation required is to change the name of the XML document to be analyzed and changing the

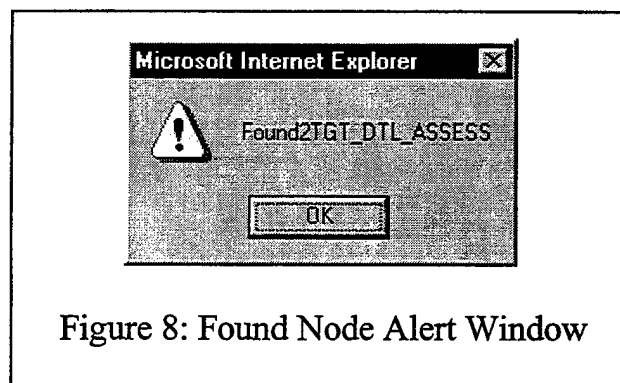


Figure 8: Found Node Alert Window

search key to "TGT_DTL_ASSESS". The corresponding alert windows are shown in the following two figures and the final output is included in Appendix G.

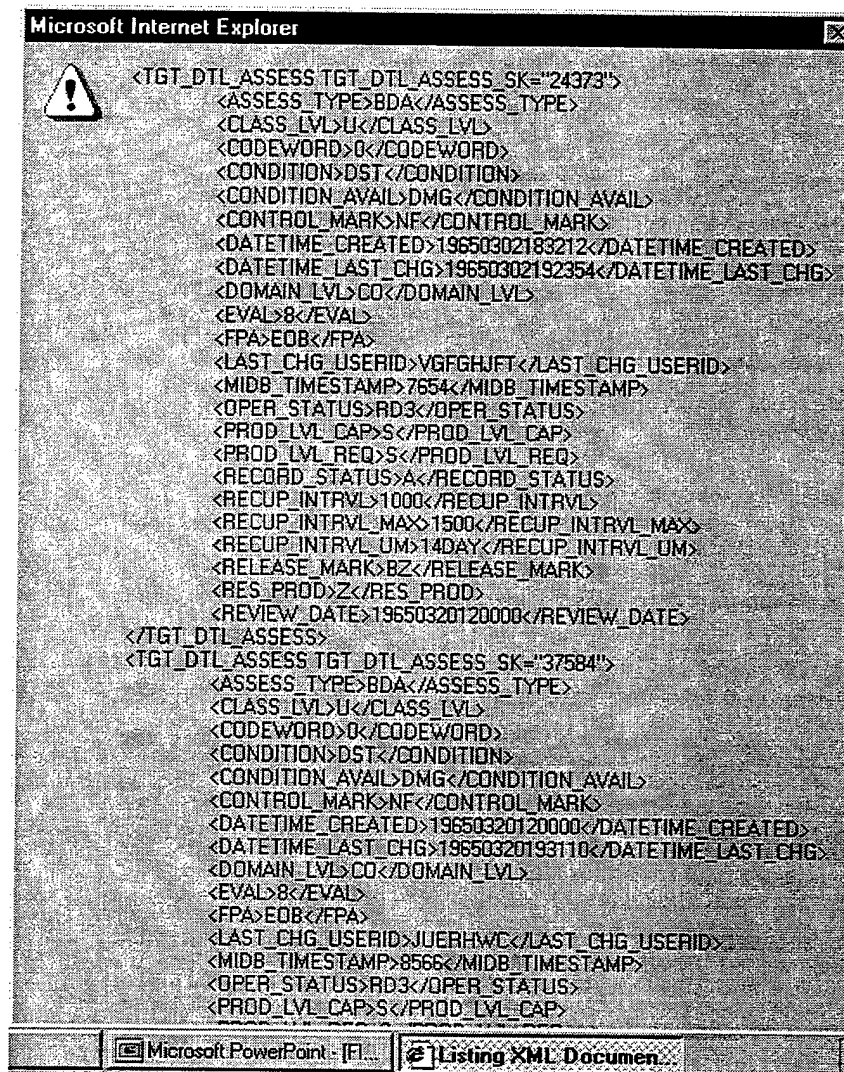


Figure 9: Snapshot of Located Nodes

G. FINAL STEP - COMPARISON OF ANALYSIS RESULTS

The objective of the previous analyses was to build a series of comparable outputs that a SME could compare to determine if the located information is common/similar. As can be seen in the outputs contained in Appendix F and Appendix G, even though the search keys used were similar, the located data looks quite different in terms of content and hierarchical relationships. This should not be unexpected when comparing heterogeneous databases. Taking into account that each database was originally developed to support a specific system-use these outputs should probably look different.

The differences in the analysis output is the very reason the analysis process was developed. It provides an alternative to the very complicated task comparing the very large databases. This analysis process has winnowed down these very large databases into two comparable XML based outputs that SMEs should be able to ascertain whether they are common.

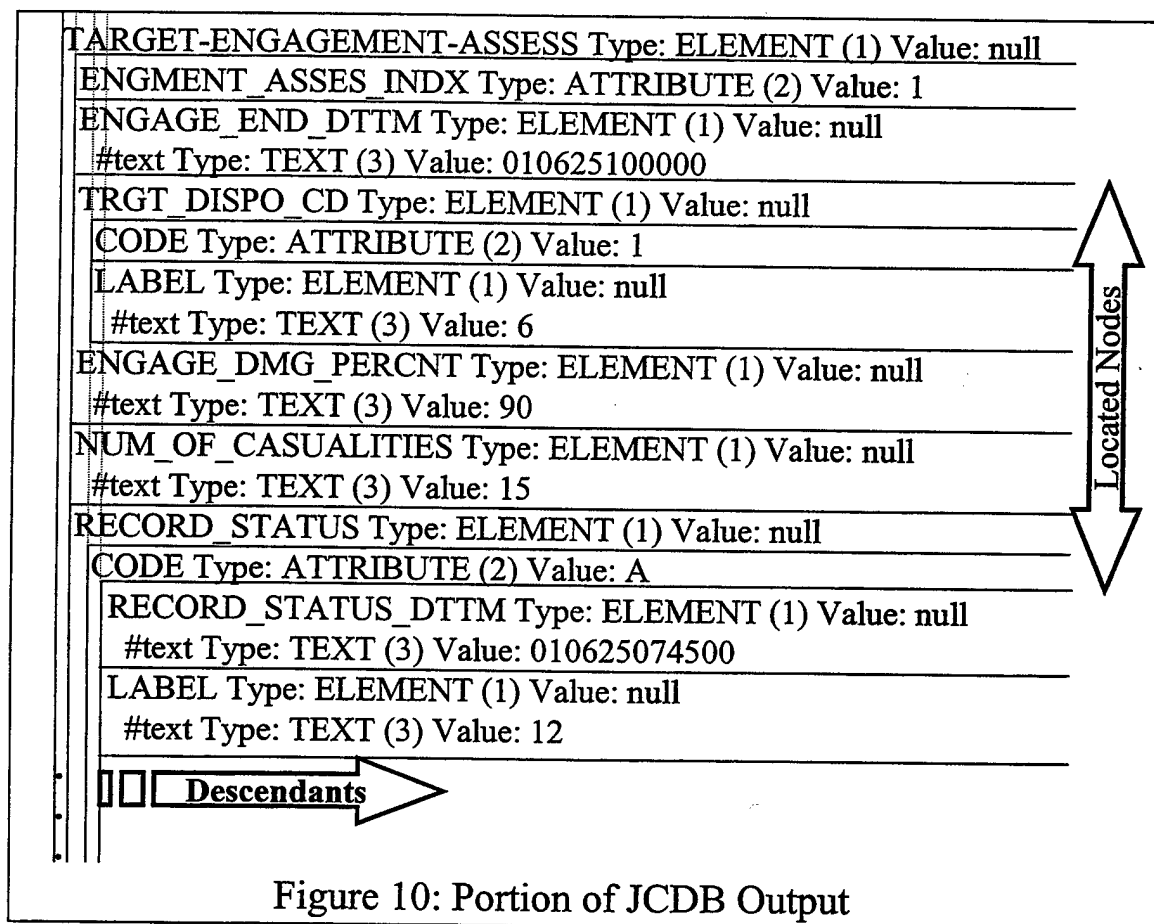
The comparison process consists of providing the SME(s) with the results of the analysis. The SME(s) can review individual components (i.e., elements, attributes, etc.) of each output along with the associated hierarchical relationships of those components. The SME can then compare

the two outputs to determine if they represent the same information.

Exactly how the outputs are compared should be left up to the individual SMEs. Different SMEs would probably emphasize different portions of the outputs during their comparisons.

1. Comparison Example

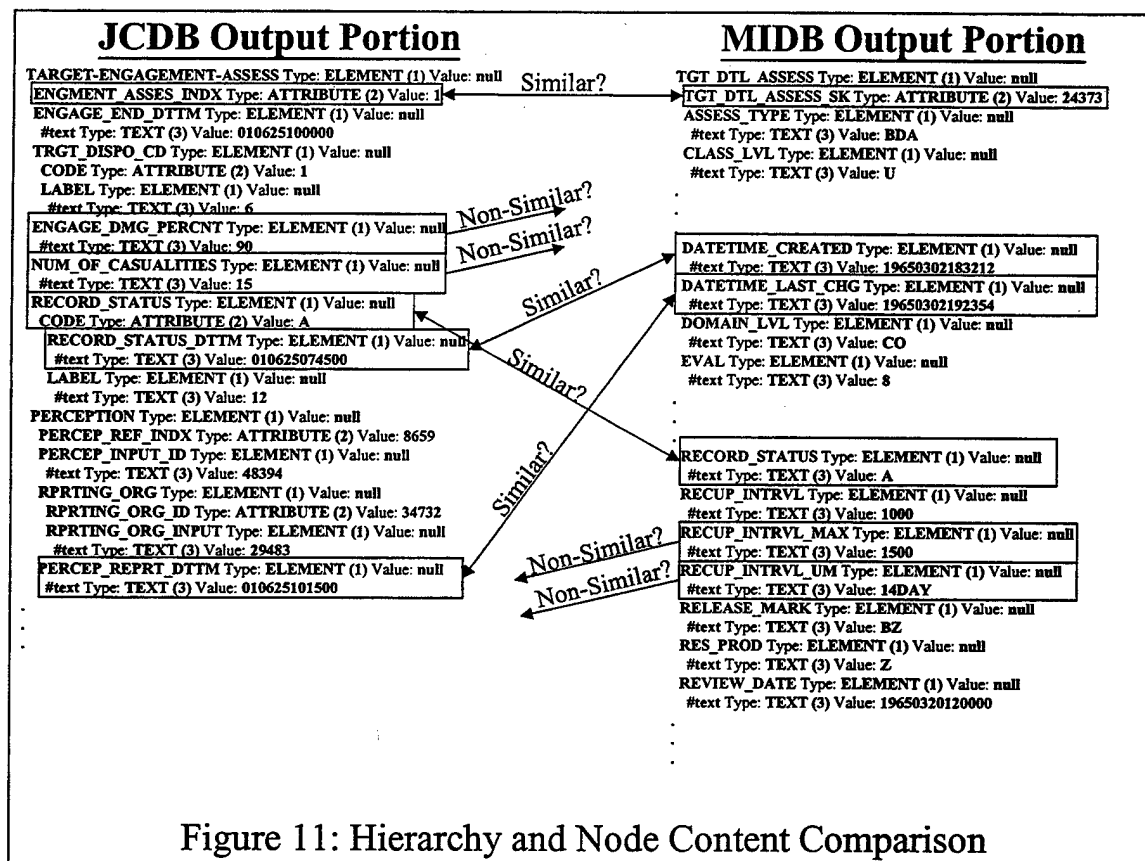
Here is an example of how an SME might compare the two extracted and decomposed outputs. Figure 10 contains a portion of the JCDB output that shows a simple overlay



method to show the hierarchical relationships and the different nodes that make up the output.

The SME could easily do this with all extracted outputs or the HTML code could be revised to show these relationships overlays. The goal would be to examine the individual nodes with regards to where they fit in the hierarchy and the type of information they contain.

Similarities between the outputs would have to be examined as well as the nodes that do not look similar. The goal would be to equate the two outputs to determine if the similar aspects of the two outputs outweigh the non-similar



aspects. As Figure 11 shows, the SME is still faced with a challenge to determine if the outputs are common.

Recalling that these two outputs were built from the previously identified common "search keys" it is interesting that when the database components and hierarchical relationships are extracted, the commonality comparison becomes a much more complex task. In fact, these extracted outputs may show that they represent very different types of information even though they were originally raised as candidates for commonality. In fact, it is the relationships between the individual elements that present the SME with the additional information necessary to execute the comparisons.

The advantage of this developed analysis and comparison method is the availability of useable information. All the necessary detail of the hierarchical node relationships and individual node content is extracted and decomposed into a series of the outputs that can be examined by the SME. It simplifies the SME's comparison tasks considerably. It also improves the thoroughness of the comparisons. Recalling that the original materials available were the databases, the data dictionaries, and the entity relationship diagrams. The database, being stored in some unique DBMS format, would be unreadable without the database software. The data dictionary and entity relationship diagrams are large, complex, and usually difficult to read.

Using this analysis and comparison method, the SME is now presented with easily readable outputs containing the desired information for comparison. If a specific term is not inherently understandable the specific portion of the data dictionary and/or entity diagrams can be consulted for detail.

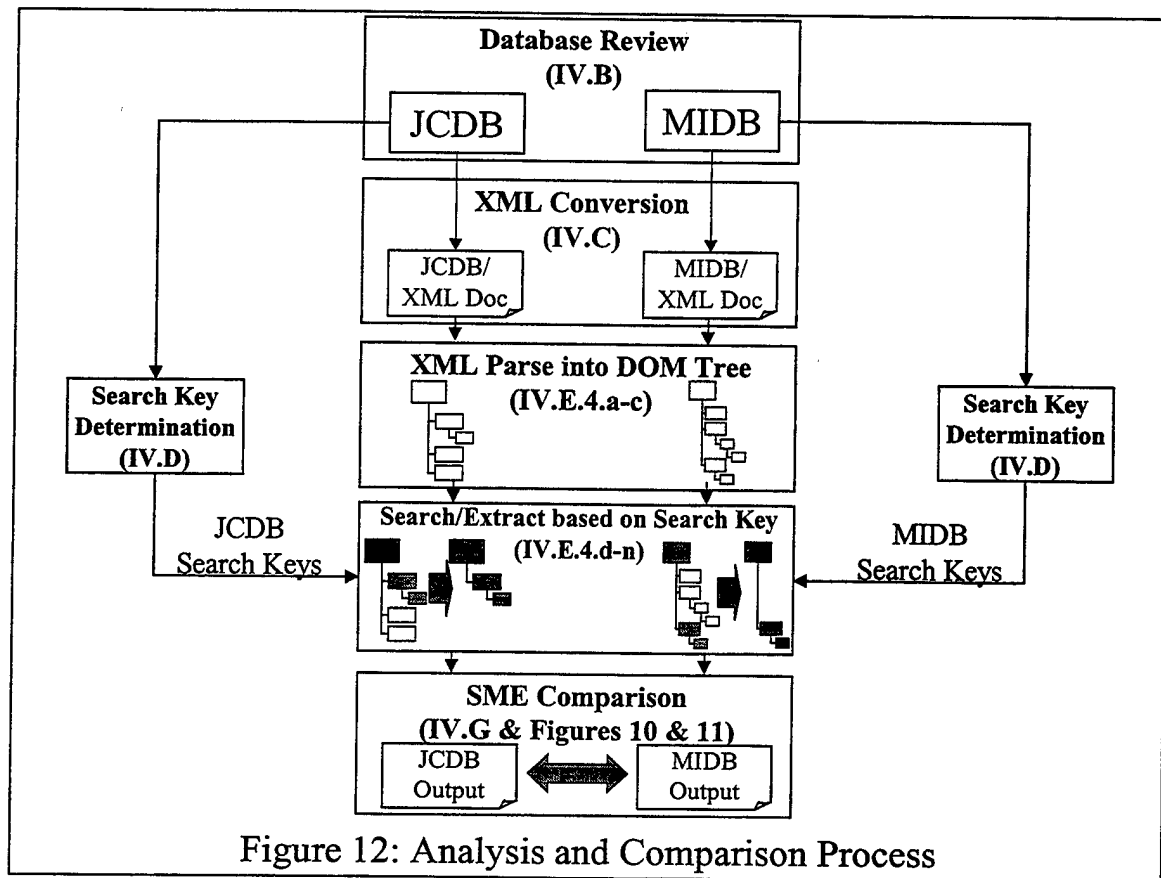
2. Comparison Summary

This completes the analysis and comparison process developed in this research effort. The goal of locating and providing potential common data from heterogeneous databases has been achieved. The data provided is in the context of both detailed information of the individual components and the hierarchical relationships of those components. Only by examining both the components and hierarchical relationships contained in the databases can the database analysis and comparison truly useful.

Hopefully, using this analysis and comparison method, the SMEs now can better focus and simplify their comparison efforts.

H. CHAPTER CONCLUSION

The database analysis and comparison process described in this chapter is represented in Figure 12. The specific sections describing each individual step is indicated by the section number included in each step. The following provides a brief summary of each step:



1. Database Review Summary

This step focuses on the manual examination of each database to be analyzed and compared. The subsequent database comparisons will be of more value if each database contains the same type of information.

2. Database Conversion to XML Summary

This step focuses on the conversion of the databases into XML documents. The recommended approach is to utilize the COTS based database to XML translators contained in most commercial DBMS. This provides an automated method to execute this task. Using automated means to complete this

task is important since the large databases, like JCDB and MIDB, will translate into extremely large XML documents.

An alternative method that can be used is to utilize the ODBC APIs built into most DBMSs to manipulate the databases into an ODBC structure. This structure can then be converted to XML by using the ODBC2XML translator shareware.

3. Entity/Attribute Analysis and Comparison Summary

This step in the process focuses on locating specific common entities and attributes between multiple databases. This provides the search keys for the subsequent hierarchical analyses of the XML documents. This process is the subject of Mr. Hamza Zobair's Thesis entitled: "An Approach for Matching Corresponding Attributes in Legacy Heterogeneous DoD Databases" [HZ01].

4. Hierarchical Examination Summary

This step is comprised of steps that parse the XML documents into DOM node trees, searches those trees using specified search keys, and extracts desired nodes and descendant nodes. This automated process, developed for this research effort, utilizes IE and its built-in XML parser to parse the XML document. Then DOM APIs are invoked using JavaScript to execute the DOM node tree search and extract the located nodes and descendant nodes. The

following table contains each of the DOM APIs and functions used to execute this search and extraction:

DOM API or Function:	Action:
getElementsByTagName(searchKey)	Builds a node list containing each tag that matches the search key.
length	Returns an integer representing the number of nodes contained in list.
cloneNode(deep)	Returns a duplicate of node. Deep is a boolean value. If false, returns node. If true, the node and entire subtree under the node is returned.
appendChild(objNode)	Appends node to existing node.
showChildNodes	A function call to examine the child nodes of the current node.
getIndent(intLevel)	A function call to improve readability of output.
nodeName	Returns name of objNode.
getNodeTypes(objNode.nodeType)	objNode.nodeType returns a number between 1 and 12.
nodeValue	Returns a text equivalent of the value contained in the node. If no value is found, null will be returned.
attributes	Generates a list of attributes contained in node.

parseError	An IE unique API that identifies any errors during XML parsing.
------------	---

5. Comparison of Analysis Results Summary

This step presents the extracted data to be compared to the SME. As can be seen in Section IV.G and Figures 10 and 11, the extracted portions can look quite different. Only a SME would be able to determine if the extracted portions are similar. When taking into account the hierarchical relationships contained in these databases, this comparison can be quite complex. During this research effort no automated methods to accomplish this task could be found that might aid the SME in this task.

The benefit this research provides is that it simplifies the SME's comparison task by providing only the desired information to be compared in an easily readable format. The SME's alternative would be to conduct exhaustive reviews of the databases, data dictionaries, and entity-relationship diagrams.

6. Research Limitations

The analysis and comparison process described contained in this thesis does have some limitations:

a) *Level of Detail*

The effort to develop databases to the same "level of detail" could be a complicated task if the databases to

be compared are significantly large and different. Databases like the ones discussed in this thesis are very large and complex. That is why smaller views of the JCDB and MIDB were used in this thesis to describe the analysis and comparison process.

b) Size Growth

Another limitation is the growth associated with the conversion of the databases to XML documents and then to DOM node trees. The size of the original databases are very large to start with. The subsequent conversions can possibly lead to a 100 to 1 growth in size. With XML being the focus of this result, this problem cannot be avoided. Fortunately, the rapidly advancing state-of-the-art in computer technology makes this less of a problem as time goes on.

c) Manual Comparison of Outputs

The SMEs are required to execute manual comparisons of the extracted portions of the databases. As shown in Figures 10 and 11 this can present the SMEs with a difficult task when considering the hierarchical node relationships and the individual node contents. The advantage provided by this process is that all the data to be compared is presented in a succinct and easy to read

format. The SME will still have to make some hard decisions on what is similar and what is not.

An alternative to the process described in this research effort is for the SMEs to rely solely on the databases, data dictionaries, and entity relationship diagrams. In that case, the comparisons would probably take days, or even weeks just to locate a single entity that is comparable in terms of hierarchy and in content.

7. Example Crosswalk

To simplify the process description provided in this chapter the following table is provided to identify the specific sections containing the individual step descriptions and associates them to the sections describing the actual step execution on the JCDB and MIDB views.

Process Step	JCDB Execution	MIDB Execution
Level of Detail IV.B	IV.B.2	IV.B.3
XML Conversion IV.C	IV.C.3	IV.C.3
Search Key Determ. IV.D	IV.D	IV.D
XML Parse to DOM IV.E.4.a-c	IV.F.2	IV.F.3
Search/Extraction IV.E.4.d-n	IV.F.2	IV.F.3

SME Comparison	IV.G	IV.G
IV.G		

8. Commercial Application of Process

The XML parse, search, and extraction portions of the described analysis and comparison process could be a useful commercial application. The code was written in a manner that would allow it to parse and analyze any XML document. It uses specified search keys to locate, extract, and output the desired nodes and descendant nodes. This code could easily be revised to take those extracted nodes and import them into new/different XML documents. This is described in greater detail in the Future Work Research Possibilities Section.

9. Putting This Research Into Practice

As described earlier in this thesis, this research is part of a larger XML-C4I Database analysis research effort. The products from this research will be incorporated with the other research efforts into an analysis process that examines multiple opportunities to use XML to facilitate manipulation of C4I databases as a means to support improved C4I interoperability.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FINAL RESEARCH POSSIBILITIES

A. CONCLUSION

This thesis examines in detail how XML can be employed to facilitate the analysis and comparison of heterogeneous databases. It provides a decomposition of located similar components that can be compared to determine what components are common and what components are not.

The first sections of this thesis provides significant detail on XML and the current DoD C4I environment. This background information provides the foundation upon which this XML based analysis and comparison process was designed, developed, executed, and described.

The original research objective being examined in this thesis was to determine if an XML schema could be defined to support the scalability of components from multiple legacy databases to modern C4I systems. This thesis successfully proves that XML based schemas can be developed that can facilitate this legacy to modern C4I migration. XML provided the common basis upon which the analysis could be executed and the common elements extracted.

This thesis describes a new XML based C4I database analysis and comparison method. It support the first step towards data exchange between C4I databases by supporting the determination of what parts of the individual C4I

databases are similar. As a result, it provides the means to facilitate the interoperability between these different C4I systems identifying data that may be exchanged by individual databases.

COTS products were employed in the development of the analysis and comparison process. By utilizing COTS an additional benefit resulting from this method was its broader application beyond C4I databases. This new method can be used to analyze and compare any XML document.

Overall, this research effort was successful in achieving its original investigation objectives.

B. FUTURE WORK RESEARCH POSSIBILITIES

Opportunities for future work in this field are extensive. The following are just a few recommendations:

- Continue to refine the analysis and comparison process developed in this research effort. As the DOM Recommendations continue to be approved, a number of new DOM APIs will become available that could be employed to improve and expand this XML based analysis and comparison process.

- Define in more detail the Subject Matter Expert (SME) comparison portion of the process. This could include an examination of how to best format the outputs from the XML based analysis to support the SME review and comparison.

- Investigate how the extracted common products from this XML based analysis and comparison could be restructured

into a global C4I database. In effect this process would reverse the process describe in this thesis. The restructuring could involve combining the extracted XML fragments into a well-formed XML document based on SME. This new XML document could be used as the basis to build future common C4I databases.

It needs to be stressed that the XML based analysis and comparison process described in this thesis describes only one way to execute an XML based database analysis. XML and XML related capabilities and tools provide the opportunity to develop alternative analysis and comparison methods. These methods could include XML related capabilities and tools like XSL, XQuery, SAX, and others. Each of these alternative methods has its own advantages and disadvantages. The use of SAX as an alternative was described in thesis.

In summary, XML provides the means to analyze and compare heterogeneous databases. This was proven through the analysis and comparison of the JCDB and MIDB Views. As XML continues to grow, so will the alternatives available to analyze heterogeneous C4I databases.

APPENDIX A - JCDB DATABASE VIEW

Example of extracted portions from data dictionary and entity relationship diagram [JDD00]:

TARGET_ENGAGEMENT_ASSESSMENT	The table that hold specifics about the results of a TARGET-ENGAGEMENT. (Battle Damage Assessment)
-------------------------------------	--

ATTRIBUTE NAME	PHYSICAL NAME	DEFINITION	DATA TYPE	NULL OPTION	ATTRIBUTE ENTITY ¹
----------------	---------------	------------	-----------	-------------	-------------------------------

TARGET-ENGAGEMENT-END index	ENGMENT ASSES_IN DX	The unique identifier that represents a specific TARGET-ENGAGEMENT-END	serial	NOT NULL	TARGET_ENGAGEMENT_ASS ESSMENT
-----------------------------	---------------------	--	--------	----------	-------------------------------

TARGET-ENGAGEMENT identifier	TARGET_ENGAGEMENT_IN DX	The unique identifier that represents a specific TARGET-ENGAGEMENT	integer	NOT NULL	TARGET_ENGAGEMENT_ASS ESSMENT
------------------------------	-------------------------	--	---------	----------	-------------------------------

TARGET_eng_input	c_INPUT	The MAC address of the machine creating the record. The unique identifier that represents a specific TARGET	integer	NOT NULL	TARGET_ENGAGEMENT_ASS ESSMENT
------------------	---------	---	---------	----------	-------------------------------

		ENGAGEMENT			
--	--	------------	--	--	--

TARGET- ENGAGEMENT-END actual end datetime	ENGAGE_E ND_DTTM	The end datetime of a TARGET engagement.	datet ime year to secon d	NOT NULL	TARGET ENGAGEM ENT ASS ESSMENT
---	---------------------	---	--	-------------	---

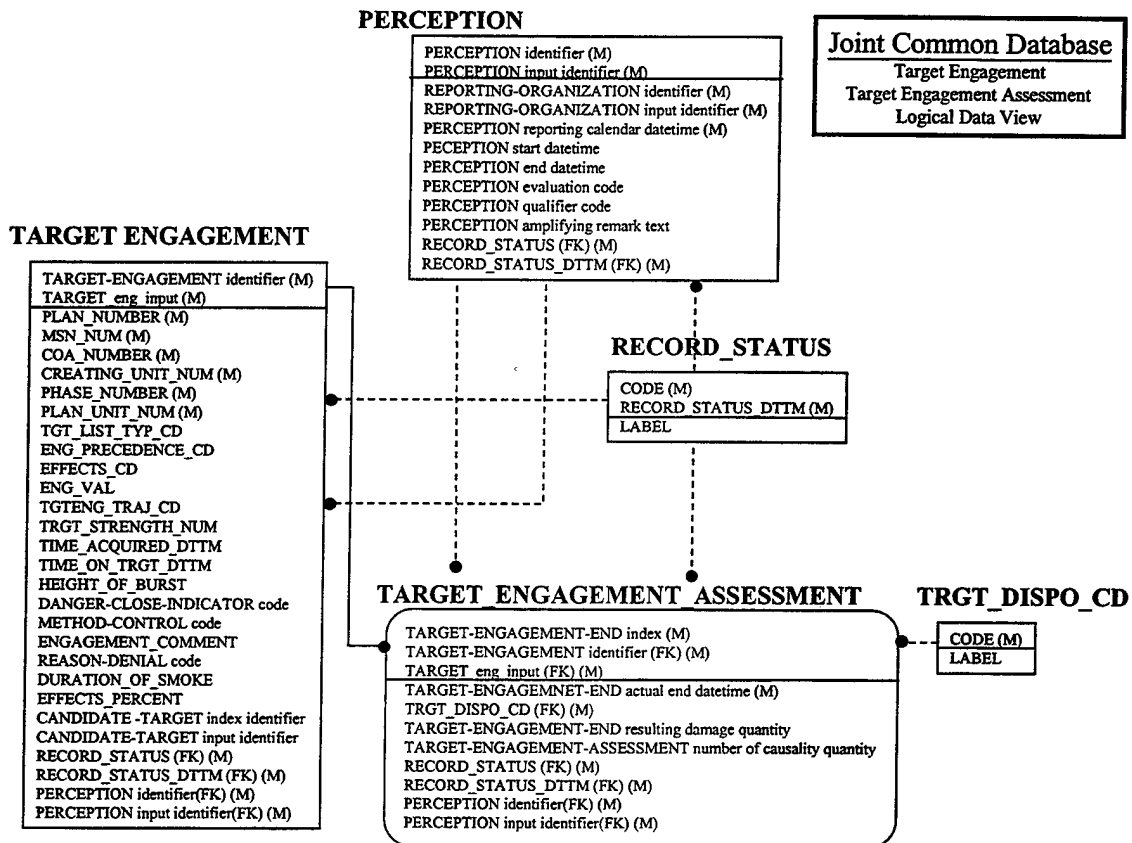
TRGT_DISPO_CD	The code which denotes the state of a TARGET after it has been ENGAGED.				
----------------------	---	--	--	--	--

CODE
LABEL

TRGT_DISPO_CD	TRGT_DIS PO_CD	The code which denotes the state of a TARGET after it has been ENGAGED.	small int	NOT NULL	TARGET ENGAGEM ENT ASS ESSMENT
---------------	-------------------	--	--------------	-------------	---

TARGET- ENGAGEMENT-END resulting damage quantity	ENGAGE_D MG_PERCN T	The percentage of destruction, or other desired result, inflicted upon a TARGET by a specific TARGET- ENGAGEMENT. (0-100%)	decim al(5, 2)	NULL	TARGET ENGAGEM ENT ASS ESSMENT
---	---------------------------	---	----------------------	------	---

TARGET- ENGAGEMENT- ASSESSMENT number of casualty quantity	NUM_OF_C ASUALITI ES	The number of casualties provided for a TARGET- ENGAGEMENT- ASSESSMENT.	integ er	NULL	TARGET ENGAGEM ENT ASS ESSMENT
---	----------------------------	--	-------------	------	---



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B - MIDB DATABASE VIEW

Example of extracted portions from data dictionary and entity relationship diagram [MID98]:

1.	Table Name:	TGT_DTL
2.	Table Long Name:	Target
3.	Description:	This table refers to a specific target. The target may be a desired mean point of impact (DMPI), or an area of impact.
4.	Elements:	
	AFFILIATION	GEOIDAL_MSL_SEPARATION
	AIR_DEF_AREA	GEOIDAL_MSL_SEPARATION_UM
	AZIMUTH	GRAPHIC_AGENCY
	AZIMUTH_REF	GRAPHIC_CC
	CC (M)	GRAPHIC_ED_DATE
	CLASS_LVL (M)	GRAPHIC_ED_NUM
	CODEWORD	GRAPHIC_SCALE
	CONDITION (M)	GRAPHIC_SERIES
	CONDITION_AVAIL	GRAPHIC_SHEET
	CONTROL_MARK	HARDNESS
	COORD (M)	HEIGHT
	COORD_BASIS (M)	HEIGHT_UM
	COORD_DATETIME	ILAT
	COORD_DATUM (M)	ILON
	COORD_DERIV (M)	JMEM_TYPE
	COORD_DERIV_ACC	LAST_CHG_USERID (M)
	COORD_DERIV_ACC_UM	LENGTH
	COORD_ROA	LENGTH_UM
	COORD_ROA_CONF_LVL	LOC_NAME
	COORD_ROA_UM	MIDB_TIMESTAMP (M)
	DATETIME_BEGIN	MIL_AREA
	DATETIME_CREATED (M)	MIL_GRID
	DATETIME_END	MIL_GRID_SYS
	DATETIME_FIRST_INFO	MSN_TYPE
	DATETIME_LAST_CHG (M)	OPER_STATUS (M)
	DATETIME_LAST_INFO	PHOTO_DATE
	DECLASS_ON	POL_SUBDIV
	DECLASS_ON_DATE	PROD_LVL_CAP (M)
	DMPI_ID	PROD_LVL_REQ (M)
	DOMAIN_LVL (M)	RADIUS
	ELEVATION	RADIUS_UM
	ELEVATION_ACC	RECORD_STATUS (M)
	ELEVATION_CONF_LVL	RECUP_INTRVL
	ELEVATION_DATUM	RECUP_INTRVL_MAX
	ELEVATION_DERIV	RECUP_INTRVL_UM
	ELEVATION_DERIV_ACC	RELEASE_MARK
	ELEVATION_DERIV_ACC_UM	RES_PROD (M)
	ELEVATION_MSL	REVIEW_DATE (M)
	ELEVATION_MSL_ACC	SHAPE
	ELEVATION_MSL_CONF_LVL	SYMBOL_CODE
	ELEVATION_MSL_DERIV	TGT_DTL_NAME (M)
	ELEVATION_MSL_DERIV_ACC	TGT_DTL_SK (M)
	ELEVATION_MSL_DERIV_ACC_UM	UTM
	ELEVATION_MSL_UM	VERTICAL_ORIENT
	ELEVATION_UM	WAC
	EVAL (M)	WATERBODY
	FPA	WIDTH
	GEODETIC_PROD	WIDTH_UM
5.	Primary Key(s):	TGT_DTL_SK
6.	Foreign Key(s):	There are none for this entity.

7. Related Table(s):
 A TGT_DTL many TGT_DTL_AIMPT_WPNs.
 A TGT_DTL many TGT_DTL_AKAs.
 A TGT_DTL many TGT_DTL_ASSESSs.
 A TGT_DTL many TGT_DTL_TIEs.
 A TGT_DTL many TGT_DTL_TIEs.
 A TGT_DTL may have many DOC_MGMT_TIEs, EQP_ELINT_MODE_TIEs, EQP_IDX_PAR_TIEs, EQP_IDX_TIEs, EQP_TIEs, EVENT_TIEs, FAC_TIEs, GEO_TIEs, IND_TIEs, NET_LINK_DTL_TIEs, NET_LINK_TIEs, NET_NODE_TIEs, OBS_TIEs, RMK_TIEs, SIG_TIEs, SOURCE_TIEs, TGT_DTL_AIMPT_WPN_TIEs, TGT_DTL_TIEs, TGT_LIST_TIEs, TGT_LIST_TIE_ORDER_TIEs, TGT_MSN_TIEs, TGT_OBJ_TIEs, TGT_SYS_TIEs, TRACK_TIEs, UNIT_ALT_LOC_TIEs, UNIT_TIE.
1. Table Name: TGT_DTL_ASSESS
2. Table Long Name: Target Assessment, Battle Damage or Strike Assessment
3. Description: This table contains information necessary for battle damage and / or strike assessment.
4. Elements:

ASSESS_DATETIME	EVAL (M)
ASSESS_TYPE (M)	FPA
CLASS_LVL (M)	LAST_CHG_USERID (M)
CODEWORD	MIDB_TIMESTAMP (M)
CONDITION (M)	OPER_STATUS (M)
CONDITION_AVAIL	PROD_LVL_CAP (M)
CONTROL_MARK	PROD_LVL_REQ (M)
DATETIME_BEGIN	RECORD_STATUS (M)
DATETIME_CREATED (M)	RECUP_INTRVL
DATETIME_END	RECUP_INTRVL_MAX
DATETIME_FIRST_INFO	RECUP_INTRVL_UM
DATETIME_LAST_CHG (M)	RELEASE_MARK
DATETIME_LAST_INFO	RES_PROD (M)
DECLASS_ON	REVIEW_DATE (M)
DECLASS_ON_DATE	TGT_DTL_ASSESS_SK (M)
DOMAIN_LVL (M)	TGT_DTL_SK (M)
5. Primary Key(s): TGT_DTL_ASSESS_SK
6. Foreign Key(s):
7. Related Table(s): TGT_DTL_SK References: TGT_DTL
7. Related Table(s):
 A TGT_DTL_ASSESS is associated with exactly one TGT_DTL.

1. Element Name: TGT_DTL_SK
2. Screen Label: Not displayed.
3. Description:
4. Structure: numeric(14,0), NOT NULL
5. Permissible Values:

SYSTEM GENERATED - SURROGATE KEY. The unique database server identifier. A numeric value, ranging from 10,000 - 99,999. The database server id will be unique for each dbserver in the MIDB worldwide network. The DB Server ID is followed by a one-up-number. A one-up-number series is maintained for each surrogate key.

6. Tables: TGT_DTL, TGT_DTL_AIMPT_WPN, TGT_DTL_AKA, TGT_DTL_ASSESS

1. Element Name: ASSESS_DATETIME
2. Screen Label: ASSESS DATETIME
3. Description: If the ASSESS_TYPE is Battle Damage Assessment (BDA) this field will contain the Time On Target value. If the ASSESS_TYPE is Strike Assessment (SA) this field will contain the Time On Target or the observation time from the report which last caused a change.
4. Structure: varchar(14), NULL
5. Permissible Values: RUL_DATETIME

[12][90][0-9][0-9]
[01-12]
[01-31]
[00-23]
[00-59]
[00-59]

Pos. 1-4, Year
Pos. 5-6, Month
Pos. 7-8, Day
Pos. 9-10, Hour
Pos. 11-12, Minute
Pos. 13-14, Second

Positions must be filled from the left. Positions on the right may be null filled. The minimum entry for this field should be a CENTURY & YEAR. As more information is available it should be filled. Conforms to the standard of ISO 8601.

6. Tables: EQP_ASSESS, FAC_ASSESS, TGT_DTL_ASSESS, TGT_SYS_ASSESS, UNIT_ASSESS, UNIT_STRIKE

-
- | | | |
|----|---------------------|--|
| 1. | Element Name: | ASSESS_TYPE |
| 2. | Screen Label: | ASSESS TYPE |
| 3. | | Description: This field indicates whether the row contains BDA or SA data. |
| 4. | Structure: | char(4), NOT NULL |
| 5. | Permissible Values: | CON_ASSESS_TYPE |
| | BDA | Battle Damage Assessment |
| | SA | Strike Assessment |
| | O | Other. Explain In Remarks. |
| | U | Unknown |
| | Z | Inconclusive Analysis |
| 6. | | Tables: EQP_ASSESS, FAC_ASSESS, TGT_DTL_ASSESS, TGT_SYS_ASSESS, UNIT_ASSESS, UNIT_STRIKE |

-
- | | | |
|----|---------------------|--|
| 1. | Element Name: | CLASS_LVL |
| 2. | Screen Label: | CLASS LVL |
| 3. | | Description: Highest classification level of the data contained within the record. |
| 4. | Structure: | char(1), NOT NULL |
| 5. | Permissible Values: | CON_CLASS_LVL |
| | U | Unclassified |
| | C | Confidential |
| | S | Secret |
| | T | Top secret |
| | Default Value: | 'S' |
| 6. | Tables: | adminsec |

Modernized Integrated Database

Target
Target Assessment/BDAR
Logical Data View

TGT_DTL

TGT_DTL_SK	
AFFILIATION	GEOIDAL_MSL_SEPARATION
AIR_DEF_AREA	GEOIDAL_MSL_SEPARATION_UM
AZIMUTH	GRAPHIC_AGENCY
AZIMUTH_REF	GRAPHIC_CC
CC (M)	GRAPHIC_ED_DATE
CLASS_LVL (M)	GRAPHIC_ED_NUM
CODEWORD	GRAPHIC_SCALE
CONDITION (M)	GRAPHIC_SERIES
CONDITION_AVAIL	GRAPHIC_SHEET
CONTROL_MARK	HARDNESS
COORD (M)	HEIGHT
COORD_BASIS (M)	HEIGHT_UM
COORD_DATETIME	ILAT
COORD_DATUM (M)	ILON
COORD_DERIV (M)	JMEM_TYPE
COORD_DERIV_ACC	LAST_CHG_USERID (M)
COORD_DERIV_ACC_UM	LENGTH
COORD_ROA	LENGTH_UM
COORD_ROA_CONF_LVL	LOC_NAME
COORD_ROA_UM	MIDB_TIMESTAMP (M)
DATETIME_BEGIN	MIL_AREA
DATETIME_CREATED (M)	MIL_GRID
DATETIME_END	MIL_GRID_SYS
DATETIME_FIRST_INFO	MSN_TYPE
DATETIME_LAST_CHG (M)	OPER_STATUS (M)
DATETIME_LAST_INFO	PHOTO_DATE
DECLASS_ON	POI_SUBDIV
DECLASS_ON_DATE	PROD_LVL_CAP (M)
DMP_ID	PROD_LVL_REQ (M)
DOMAIN_LVL (M)	RADIUS
ELEVATION	RADIUS_UM
ELEVATION_ACC	RECORD_STATUS (M)
ELEVATION_CONF_LVL	RECUPT_INTRVL
ELEVATION_DATUM	RECUPT_INTRVL_MAX
ELEVATION_DERIV	RECUPT_INTRVL_UM
ELEVATION_DERIV_ACC	RELEASE_MARK
ELEVATION_DERIV_ACC_UM	RES_PROD (M)
ELEVATION_MSL	REVIEW_DATE (M)
ELEVATION_MSL_ACC	SHAPE
ELEVATION_MSL_CONF_LVL	SYMBOL_CODE
ELEVATION_MSL_DERIV	TGT_DTL_NAME (M)
ELEVATION_MSL_DERIV_ACC	TGT_DTL_SK (M)
ELEVATION_MSL_DERIV_ACC_UM	UTM
ELEVATION_MSL_UM	VERTICAL_ORIENT
ELEVATION_UM	WAC
EVAL (M)	WATERBODY
FPA	WIDTH
GEODETTIC_PROD	WIDTH_UM

TGT_DTL_AKA

TGT_DTL_AKA_SK	
AKA (M)	DOMAIN_LVL (M)
AKA_TYPE (M)	EVAL (M)
CLASS_LVL (M)	FPA
CODEWORD	LAST_CHG_USERID (M)
CONTROL_MARK	MIDB_TIMESTAMP (M)
DATETIME_BEGIN	PROD_LVL_CAP (M)
DATETIME_CREATED (M)	PROD_LVL_REQ (M)
DATETIME_END	RECORD_STATUS (M)
DATETIME_FIRST_INFO	RELEASE_MARK
DATETIME_LAST_CHG (M)	RES_PROD (M)
DATETIME_LAST_INFO	REVIEW_DATE (M)
DECLASS_ON	TGT_DTL_AKA_SK (M)
DECLASS_ON_DATE	TGT_DTL_SK (M)

TGT_DTL_ASSESS

TGT_DTL_ASSESS_SK	
ASSESS_DATETIME	EVAL (M)
ASSESS_TYPE (M)	FPA
CLASS_LVL (M)	LAST_CHG_USERID (M)
CODEWORD	MIDB_TIMESTAMP (M)
CONDITION (M)	OPER_STATUS (M)
CONDITION_AVAIL	PROD_LVL_CAP (M)
CONTROL_MARK	PROD_LVL_REQ (M)
DATETIME_BEGIN	RECORD_STATUS (M)
DATETIME_CREATED (M)	RECUPT_INTRVL
DATETIME_END	RECUPT_INTRVL_MAX
DATETIME_FIRST_INFO	RECUPT_INTRVL_UM
DATETIME_LAST_CHG (M)	RELEASE_MARK
DATETIME_LAST_INFO	RES_PROD (M)
DECLASS_ON	REVIEW_DATE (M)
DECLASS_ON_DATE	TGT_DTL_ASSESS_SK (M)
DOMAIN_LVL (M)	TGT_DTL_SK (M)

TGT_DTL_TIE

TGT_DTL_TIE_SK	
ASSOC (M)	EVAL (M)
ASSOC_BEGIN_DATE	FPA
ASSOC_END_DATE	LAST_CHG_USERID (M)
CLASS_LVL (M)	MIDB_TIMESTAMP (M)
CODEWORD	PROD_LVL_CAP (M)
CONTROL_MARK	PROD_LVL_REQ (M)
DATETIME_BEGIN	RECORD_STATUS (M)
DATETIME_CREATED (M)	RELEASE_MARK
DATETIME_END	RES_PROD (M)
DATETIME_FIRST_INFO	REVIEW_DATE (M)
DATETIME_LAST_CHG (M)	TGT_DTL_TIE_SK (M)
DATETIME_LAST_INFO	TIE_BOOL (M)
DECLASS_ON	TIE_FROM_SK (M)
DECLASS_ON_DATE	TIE_TO_ENTITY (M)
DOMAIN_LVL (M)	TIE_TO_SK (M)

APPENDIX C - JCDB XML DOCUMENT

```

<?xml version="1.0"?>
<TARGET-ENGAGEMENT TARGET_ENGAGE_INDEX="5832">
  <c_INPUT>184</c_INPUT>
  <PLAN_NUMBER>67398</PLAN_NUMBER>
  <MSN_NUMBER>16041</MSN_NUMBER>
  <COA_NUMBER>8</COA_NUMBER>
  <CREATING_UNIT_NUM>21</CREATING_UNIT_NUM>
  <PHASE_NUMBER>5</PHASE_NUMBER>
  <PLAN_UNIT_NUMBER>2</PLAN_UNIT_NUMBER>
  <TGT_LIST_TYP_CD>19</TGT_LIST_TYP_CD>
  <ENG_PRECEDENCE_CD>1</ENG_PRECEDENCE_CD>
  <EFFECTS_CD>3</EFFECTS_CD>
  <TRGT_STRENGTH_NUM>10</TRGT_STRENGTH_NUM>
  <TIME_ACQUIRED_DTTM>010625085623</TIME_ACQUIRED_DTTM>
  <TIME_ON_TRGT_DTTM>010625100000</TIME_ON_TRGT_DTTM>
  <ENGAGEMENT_COMMENT>ENGAGE COORDINATED W/1ST
CAV</ENGAGEMENT_COMMENT>
  <EFFECTS_PERCENT>75</EFFECTS_PERCENT>
  <RECORD_STATUS_CODE="A">
    <RECORD_STATUS_DTTM>010625074500</RECORD_STATUS_DTTM>
    <LABEL>12</LABEL>
  </RECORD_STATUS>
  <PERCEPTION PERCEP_REF_INDX="8659">
    <PERCEP_INPUT_ID>48394</PERCEP_INPUT_ID>
    <RPRTING_ORG RPRTING_ORG_ID="34732">
      <RPRTING_ORG_INPUT>29483</RPRTING_ORG_INPUT>
    </RPRTING_ORG>
    <PERCEP_REPRT_DTTM>010625091500</PERCEP_REPRT_DTTM>
    <PERCEP_STRT_DTTM>010625085500</PERCEP_STRT_DTTM>
    <PERCEP_END_DTTM>010625090000</PERCEP_END_DTTM>
    <RECORD_STATUS_CODE="A">
      <RECORD_STATUS_DTTM>010625091000</RECORD_STATUS_DTTM>
      <LABEL>10</LABEL>
    </RECORD_STATUS>
  </PERCEPTION>
  <TARGET-ENGAGEMENT-ASSESS ENGMENT_ASSES_INDX="1">
    <ENGAGE_END_DTTM>010625100000</ENGAGE_END_DTTM>
    <TRGT_DISPO_CD CODE="1">
      <LABEL>6</LABEL>
    </TRGT_DISPO_CD>
    <ENGAGE_DMG_PERCNT>90</ENGAGE_DMG_PERCNT>
    <NUM_OF_CASUALITIES>15</NUM_OF_CASUALITIES>
    <RECORD_STATUS_CODE="A">
      <RECORD_STATUS_DTTM>010625074500</RECORD_STATUS_DTTM>
      <LABEL>12</LABEL>
    </RECORD_STATUS>
  </TARGET-ENGAGEMENT-ASSESS>
</TARGET-ENGAGEMENT>

```

```

<PERCEPTION PERCEP_REF_INDX="8659">
  <PERCEP_INPUT_ID>48394</PERCEP_INPUT_ID>
  <RPRTING_ORG RPRTING_ORG_ID="34732">
    <RPRTING_ORG_INPUT>29483</RPRTING_ORG_INPUT>
  </RPRTING_ORG>
  <PERCEP_REPRT_DTTM>010625101500</PERCEP_REPRT_DTTM>
  <PERCEP_STRT_DTTM>010625100000</PERCEP_STRT_DTTM>
  <PERCEP_END_DTTM>010625100500</PERCEP_END_DTTM>
  <RECORD_STATUS_CODE="A">
    <RECORD_STATUS_DTTM>010625101500</RECORD_STATUS_DTTM>
    <LABEL>10</LABEL>
  </RECORD_STATUS>
</PERCEPTION>
</TARGET-ENGAGEMENT-ASSESS>
<TARGET-ENGAGEMENT-ASSESS ENGMT_ASSES_INDX="2">
  <ENGAGE_END_DTTM>010625121500</ENGAGE_END_DTTM>
  <TRGT_DISPO_CD_CODE="1">
    <LABEL>6</LABEL>
  </TRGT_DISPO_CD>
  <ENGAGE_DMG_PERCNT>80</ENGAGE_DMG_PERCNT>
  <NUM_OF_CASUALITIES>10</NUM_OF_CASUALITIES>
  <RECORD_STATUS_CODE="A">
    <RECORD_STATUS_DTTM>010625123000</RECORD_STATUS_DTTM>
    <LABEL>12</LABEL>
  </RECORD_STATUS>
  <PERCEPTION PERCEP_REF_INDX="8659">
    <PERCEP_INPUT_ID>48394</PERCEP_INPUT_ID>
    <RPRTING_ORG RPRTING_ORG_ID="34732">
      <RPRTING_ORG_INPUT>29483</RPRTING_ORG_INPUT>
    </RPRTING_ORG>
    <PERCEP_REPRT_DTTM>010625124000</PERCEP_REPRT_DTTM>
    <PERCEP_STRT_DTTM>010625122000</PERCEP_STRT_DTTM>
    <PERCEP_END_DTTM>010625123000</PERCEP_END_DTTM>
    <RECORD_STATUS_CODE="A">
      <RECORD_STATUS_DTTM>010625124000</RECORD_STATUS_DTTM>
      <LABEL>10</LABEL>
    </RECORD_STATUS>
  </PERCEPTION>
</TARGET-ENGAGEMENT-ASSESS>
</TARGET-ENGAGEMENT>

```

APPENDIX D - MIDB XML DOCUMENT

```

<?xml version="1.0"?>
<TARGET TGT_DTL_SK="19954">
  <AFFILIATION>H</AFFILIATION>
  <COUNTRY>IQ</COUNTRY>
  <CLASS_LVL>U</CLASS_LVL>
  <CONDITION>CCD</CONDITION>
  <COORD>234853658S1453834674W</COORD>
  <COORD_BASIS>GA</COORD_BASIS>
  <COORD_DATUM>BUP</COORD_DATUM>
  <COORD_DERIV>K</COORD_DERIV>
  <DATETIME_CREATED>19650229122543</DATETIME_CREATED>
  <DATETIME_LAST_CHG>19650229161445</DATETIME_LAST_CHG>
  <DOMAIN_LVL>CO</DOMAIN_LVL>
  <EVAL>1</EVAL>
  <LAST_CHG_USERID>DJFGEIDG</LAST_CHG_USERID>
  <MIDB_TIMESTAMP>2347</MIDB_TIMESTAMP>
  <OPER_STATUS>RD0</OPER_STATUS>
  <PROD_LVL_CAP>S</PROD_LVL_CAP>
  <PROD_LVL_REQ>S</PROD_LVL_REQ>
  <RECORD_STATUS>A</RECORD_STATUS>
  <RES_PROD>XX</RES_PROD>
  <REVIEW_DATE>19650229170210</REVIEW_DATE>
  <TGT_DTL_NAME>BADGUYS</TGT_DTL_NAME>
  <TGT_DTL_AKA TGT_DTL_AKA_SK="87442">
    <AKA>REALLYBADGUYS</AKA>
    <AKA_TYPE>OAP</AKA_TYPE>
    <CLASS_LVL>U</CLASS_LVL>
    <DATETIME_CREATED>19650301063043</DATETIME_CREATED>
    <DATETIME_LAST_CHG>19650301065545</DATETIME_LAST_CHG>
    <DOMAIN_LVL>CO</DOMAIN_LVL>
    <EVAL>1</EVAL>
    <LAST_CHG_USERID>HJGFGYVT</LAST_CHG_USERID>
    <MIDB_TIMESTAMP>4345</MIDB_TIMESTAMP>
    <OPER_STATUS>RD0</OPER_STATUS>
    <PROD_LVL_CAP>S</PROD_LVL_CAP>
    <PROD_LVL_REQ>S</PROD_LVL_REQ>
    <RECORD_STATUS>A</RECORD_STATUS>
    <RES_PROD>XX</RES_PROD>
    <REVIEW_DATE>19650310120000</REVIEW_DATE>
  </TGT_DTL_AKA>
  <TGT_DTL_ASSESS TGT_DTL_ASSESS_SK="24373">
    <ASSESS_TYPE>BDA</ASSESS_TYPE>
    <CLASS_LVL>U</CLASS_LVL>
    <CODEWORD>0</CODEWORD>
    <CONDITION>DST</CONDITION>
    <CONDITION_AVAIL>DMG</CONDITION_AVAIL>

```

```

<CONTROL_MARK>NF</CONTROL_MARK>
<DATETIME_CREATED>19650302183212</DATETIME_CREATED>
<DATETIME_LAST_CHG>19650302192354</DATETIME_LAST_CHG>
<DOMAIN_LVL>CO</DOMAIN_LVL>
<EVAL>8</EVAL>
<FPA>EOB</FPA>
<LAST_CHG_USERID>VGFGHJFT</LAST_CHG_USERID>
<MIDB_TIMESTAMP>7654</MIDB_TIMESTAMP>
<OPER_STATUS>RD3</OPER_STATUS>
<PROD_LVL_CAP>S</PROD_LVL_CAP>
<PROD_LVL_REQ>S</PROD_LVL_REQ>
<RECORD_STATUS>A</RECORD_STATUS>
<RECUP_INTRVL>1000</RECUP_INTRVL>
<RECUP_INTRVL_MAX>1500</RECUP_INTRVL_MAX>
<RECUP_INTRVL_UM>14DAY</RECUP_INTRVL_UM>
<RELEASE_MARK>BZ</RELEASE_MARK>
<RES_PROD>Z</RES_PROD>
<REVIEW_DATE>19650320120000</REVIEW_DATE>
</TGT_DTL_ASSESS>
<TGT_DTL_ASSESS TGT_DTL_ASSESS_SK="37584">
  <ASSESS_TYPE>BDA</ASSESS_TYPE>
  <CLASS_LVL>U</CLASS_LVL>
  <CODEWORD>0</CODEWORD>
  <CONDITION>DST</CONDITION>
  <CONDITION_AVAIL>DMG</CONDITION_AVAIL>
  <CONTROL_MARK>NF</CONTROL_MARK>
  <DATETIME_CREATED>19650320120000</DATETIME_CREATED>
  <DATETIME_LAST_CHG>19650320193110</DATETIME_LAST_CHG>
  <DOMAIN_LVL>CO</DOMAIN_LVL>
  <EVAL>8</EVAL>
  <FPA>EOB</FPA>
  <LAST_CHG_USERID>JUERHWC</LAST_CHG_USERID>
  <MIDB_TIMESTAMP>8566</MIDB_TIMESTAMP>
  <OPER_STATUS>RD3</OPER_STATUS>
  <PROD_LVL_CAP>S</PROD_LVL_CAP>
  <PROD_LVL_REQ>S</PROD_LVL_REQ>
  <RECORD_STATUS>A</RECORD_STATUS>
  <RECUP_INTRVL>2500</RECUP_INTRVL>
  <RECUP_INTRVL_MAX>5000</RECUP_INTRVL_MAX>
  <RECUP_INTRVL_UM>30DAY</RECUP_INTRVL_UM>
  <RELEASE_MARK>BZ</RELEASE_MARK>
  <RES_PROD>Z</RES_PROD>
  <REVIEW_DATE>19650322120000</REVIEW_DATE>
</TGT_DTL_ASSESS>
<TGT_DTL_TIE TGT_DTL_TIE_SK="34578">
  <ASSOC>AZ</ASSOC>
  <CLASS_LVL>U</CLASS_LVL>
  <DATETIME_CREATED>19650229150150</DATETIME_CREATED>
  <DATETIME_LAST_CHG>19650229161212</DATETIME_LAST_CHG>
  <DOMAIN_LVL>CO</DOMAIN_LVL>
  <EVAL>1</EVAL>

```

<LAST_CHG_USERID>BNVVTf</LAST_CHG_USERID>
<MIDB_TIMESTAMP>9879</MIDB_TIMESTAMP>
<OPER_STATUS>RD0</OPER_STATUS>
<PROD_LVL_CAP>S</PROD_LVL_CAP>
<PROD_LVL_REQ>S</PROD_LVL_REQ>
<RECORD_STATUS>A</RECORD_STATUS>
<RES_PROD>XX</RES_PROD>
<REVIEW_DATE>19650315120000</REVIEW_DATE>
<TIE_BOOL>0</TIE_BOOL>
<TIE_FROM_SK>23429</TIE_FROM_SK>
<TIE_TO_ENTITY>TRG_DTL</TIE_TO_ENTITY>
</TGT_DTL_TIE>
</TARGET>

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E - ANALYSIS AND MANIPULATION CODE

```
<html>
<head>
<!-- Code adapted from code provided in XML IE5 (see references). →
<title>Listing XML Document Nodes and Attributes</title>
<style type="text/css">
BODY {font-family:Tahoma,Verdana,Arial,sans-serif; font-size:12px; font-
weight:normal}
.intro {font-family:Tahoma,Verdana,Arial,sans-serif; font-size:14px;
font-weight:bold}
</style>

<!-- The SRC will have to be changed to name of XML file to be searched.
→
<XML ID="domSearchList" SRC="jcdbxml.xml"></XML>

<!-- Code built in JavaScript →
<SCRIPT LANGUAGE="JavaScript">

<!--
//Function parses and checks for errors.
//Note the parseError is a Microsoft extension to the W3C DOM.
//It is the only Microsoft extension used in this code.
//All other APIs used are included in W3C DOM Recommendation #1.
function parseXML() {
//Develops DOM from XML.
objXMLData = document.all['domSearchList'];
if (objXMLData.parseError.errorCode != 0) {
    alert('Invalid XML file: ' + objXMLData.parseError.reason);
    return;
}

//Modify "XXXXXX" in searchDocument(objXMLData, "XXXXXX")
//to desired search key. Note: Case sensitivity is important!!
divResults.innerHTML = searchDocument(objXMLData, "TARGET-ENGAGEMENT-
ASSESS");
}

//Primary function that executes search of DOM
function searchDocument(sourceDoc, searchKey) {
// declare local variables
var objNode;
var strNodes = '';
var I = 0;
var listNodes = '';
var foundNode = '';
var clonedNode;

//Creates XML DOM Fragment.
```

```

var objFrag;
objFrag = sourceDoc.createDocumentFragment();

// Finds the root of XML document.
theRoot = sourceDoc.documentElement;

// Search for authors.
// Develops a list of Nodes matching search key.
listNodes = theRoot.getElementsByTagName(searchKey);

// Check and show alert on how many found.
// listNodes.length provides number of nodes in list.
if (listNodes.length > 0) {
    alert ('Found' + listNodes.length + searchKey);
}
else {
    alert ('Found' + listNodes.length + searchKey);
    alert ('Check and re-enter Search Key. Note case sensitivity is
important!!!');
}
// Loops through list of authors.
for (I = 0; I < listNodes.length; I++)
{
    //Sets located node.
    foundNode = listNodes(I);

    //Clones (copies) node ("true" attribute results in all children
being cloned).
    objNode = foundNode.cloneNode(true);

    //Appends cloned node (and children) to XML fragment)
    objFrag.appendChild(objNode);

    //Loop repeats until nodeList is exhausted.
}
//Provides quick output of built XML fragment.
alert (objFrag.xml);

// Loop through list of authors again to breakdown fragment and then
build output.
for (I = 0; I < listNodes.length; I++)
{
    foundNode = listNodes(I);

    //strNode is simple text variable that is continuously appended to
build output.
    //Calls showChildNodes function that decomposes node.
    //Also checks for attributes and checks for child nodes.
    //If child nodes exist, the showChildNodes calls the showsChildNodes
function again.
    //This recursive function call continues until no are children left.

```

```

    //Then returns strNode.
    strNodes += showChildNodes(foundNode, 0);
    strNodes += '</B><BR>' + '</B><BR>';
    }
    return strNodes;
}

//objNode provides the top level foundNode that is to be decomposed.
//intLevel provide the indentation detail as strNode is being built.
function showChildNodes(objNode, intLevel) {
    var strNodes = '';
    var intCount = 0;
    var intNode = 0;

    // Gets the values for this node.
    strNodes += getIndent(intLevel) + '<B>' + objNode.nodeName
        + '</B> &nbsp; Type: <B>' + getNodeTypes(objNode.nodeType)
        + '</B> &nbsp; Value: <B>' + objNode.nodeValue + '</B><BR>';

    // Checks for any attributes.
    objAttrList = objNode.attributes;
    if (objAttrList != null) {
        intCount = objAttrList.length;
        if (intCount > 0) {
            // For each attribute, displays the attribute information.
            for (intAttr = 0; intAttr < intCount; intAttr++) {
                strNodes += getIndent(intLevel + 1) + '<B>'
                    + objAttrList(intAttr).nodeName + '</B> &nbsp; Type:
<B>'
                    + getNodeTypes(objAttrList(intAttr).nodeType)
                    + '</B> &nbsp; Value: <B>'
                    + objAttrList(intAttr).nodeValue + '</B><BR>';
            }
        }
    }

    // Checks for any child nodes.
    intCount = objNode.childNodes.length;
    if (intCount > 0) {
        // For each child node, display the node, attributes and its child
        node information.
        for (intNode = 0; intNode < intCount; intNode++) {

            //Recursive showChildNodes function call.
            strNodes += showChildNodes(objNode.childNodes(intNode), intLevel +
1);
        }
    }
    return strNodes;
}

```



```

//→
</SCRIPT>

</head>
<!-- HTML is simply used to facilitate calling of functions →
<!-- and output of strNode →
<body BGCOLOR="#FFFFFF" ONLOAD="parseXML()">
<SPAN CLASS=intro>Located and Decomposed Elements and Children</SPAN><P>

<!-- to insert the results of parsing the object model →
<DIV ID="divResults">Parsing XML file ...</DIV>

<!-- Button function below displays original XML file being examined. →
<!-- Will have to change name of XML file to one to be examined. →
<!-- Make sure XML file is contained in same folder as this file. →
<!-- CAUTION CAUTION CAUTION →
<!-- If examining extremely large XML files built from large databases
→
<!-- like JCDB or MIDB. →
<!-- Best to change Button call to a comment line like the one below →

<!-- <INPUT TYPE="BUTTON" VALUE="&nbsp; &nbsp; &nbsp;"
ONCLICK="location.href='onebook.xml'">
&nbsp;Display the XML →

<HR><B>&nbsp; &nbsp; &nbsp;
<INPUT TYPE="BUTTON" VALUE="&nbsp; &nbsp; &nbsp;"
ONCLICK="location.href='JCDBXML.xml'">
&nbsp;Display the XML
</B><P>

</body>
</html>

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F - JCDB ANALYSIS OUTPUT

Located and Decomposed Elements and Children

TARGET-ENGAGEMENT-ASSESS Type: ELEMENT (1) Value: null
ENGMENT_ASSES_INDX Type: ATTRIBUTE (2) Value: 1
ENGAGE_END_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625100000
TRGT_DISPO_CD Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: 1
LABEL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 6
ENGAGE_DMG_PERCNT Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 90
NUM_OF_CASUALITIES Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 15
RECORD_STATUS Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: A
RECORD_STATUS_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625074500
LABEL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 12
PERCEPTION Type: ELEMENT (1) Value: null
PERCEP_REF_INDX Type: ATTRIBUTE (2) Value: 8659
PERCEP_INPUT_ID Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 48394
RPRTING_ORG Type: ELEMENT (1) Value: null
RPRTING_ORG_ID Type: ATTRIBUTE (2) Value: 34732
RPRTING_ORG_INPUT Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 29483
PERCEP_REPRT_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625101500
PERCEP_STRT_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625100000
PERCEP_END_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625100500
RECORD_STATUS Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: A
RECORD_STATUS_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625101500
LABEL Type: ELEMENT (1) Value: null

#text Type: TEXT (3) Value: 10

TARGET-ENGAGEMENT-ASSESS Type: ELEMENT (1) Value: null
ENGMENT_ASSES_INDX Type: ATTRIBUTE (2) Value: 2
ENGAGE_END_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625121500
TRGT_DISPO_CD Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: 1
LABEL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 6
ENGAGE_DMG_PERCNT Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 80
NUM_OF_CASUALITIES Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 10
RECORD_STATUS Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: A
RECORD_STATUS_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625123000
LABEL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 12
PERCEPTION Type: ELEMENT (1) Value: null
PERCEP_REF_INDX Type: ATTRIBUTE (2) Value: 8659
PERCEP_INPUT_ID Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 48394
RPRTING_ORG Type: ELEMENT (1) Value: null
RPRTING_ORG_ID Type: ATTRIBUTE (2) Value: 34732
RPRTING_ORG_INPUT Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 29483
PERCEP_REPRT_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625124000
PERCEP_STRT_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625122000
PERCEP_END_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625123000
RECORD_STATUS Type: ELEMENT (1) Value: null
CODE Type: ATTRIBUTE (2) Value: A
RECORD_STATUS_DTTM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 010625124000
LABEL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 10

APPENDIX G - MIDB ANALYSIS OUTPUT

Located and Decomposed Elements and Children

TGT_DTL_ASSESS Type: ELEMENT (1) Value: null
TGT_DTL_ASSESS_SK Type: ATTRIBUTE (2) Value: 24373
ASSESS_TYPE Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: BDA
CLASS_LVL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: U
CODEWORD Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 0
CONDITION Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: DST
CONDITION_AVAIL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: DMG
CONTROL_MARK Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: NF
DATETIME_CREATED Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 19650302183212
DATETIME_LAST_CHG Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 19650302192354
DOMAIN_LVL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: CO
EVAL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 8
FPA Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: EOB
LAST_CHG_USERID Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: VGFGHJFT
MIDB_TIMESTAMP Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 7654
OPER_STATUS Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: RD3
PROD_LVL_CAP Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: S
PROD_LVL_REQ Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: S
RECORD_STATUS Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: A
RECUP_INTRVL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 1000
RECUP_INTRVL_MAX Type: ELEMENT (1) Value: null

#text Type: TEXT (3) Value: 1500
 RECUP_INTRVL_UM Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 14DAY
 RELEASE_MARK Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: BZ
 RES_PROD Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: Z
 REVIEW_DATE Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 19650320120000

 TGT_DTL_ASSESS Type: ELEMENT (1) Value: null
 TGT_DTL_ASSESS_SK Type: ATTRIBUTE (2) Value: 37584
 ASSESS_TYPE Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: BDA
 CLASS_LVL Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: U
 CODEWORD Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 0
 CONDITION Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: DST
 CONDITION_AVAIL Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: DMG
 CONTROL_MARK Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: NF
 DATETIME_CREATED Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 19650320120000
 DATETIME_LAST_CHG Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 19650320193110
 DOMAIN_LVL Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: CO
 EVAL Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 8
 FPA Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: EOB
 LAST_CHG_USERID Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: JUERHWC
 MIDB_TIMESTAMP Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: 8566
 OPER_STATUS Type: ELEMENT (1) Value: null
 #text Type: TEXT (3) Value: RD3
 PROD_LVL_CAP Type: ELEMENT (1) Value: null

#text Type: TEXT (3) Value: S
PROD_LVL_REQ Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: S
RECORD_STATUS Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: A
RECUP_INTRVL Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 2500
RECUP_INTRVL_MAX Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 5000
RECUP_INTRVL_UM Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 30DAY
RELEASE_MARK Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: BZ
RES_PROD Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: Z
REVIEW_DATE Type: ELEMENT (1) Value: null
#text Type: TEXT (3) Value: 19650322120000

THIS PAGE INTENTIONALLY LEFT BLANK

GLOSSARY

ABCS	Army Battlefield Command System
AFATDS	The Advanced Field Artillery Tactical Data System
API	Application Programming Interface
BDAR	Battle Damage Assessment Report
C2	Command and Control
C4I	Command, Control, Communication, Computers, and Intelligence
COI	Community of Interest
COP	Common Operational Picture
COTS	Commercial-Off-The-Shelf
CTP	Common Tactical Picture
DBMS	Database Management System
DIICOE	Defense Information Infrastructure Common Operating Environment
DISA	Defense Information Systems Agency
DoD	Department of Defense
DOM	Document Object Model
DTD	Document Type Definition
GCCS	Global Command and Control System
GML	Generalized Markup Language
HTML	Hypertext Markup Language
I ³	Integrated, Imagery and Intelligence
IDEF1X	Integration Definition for Information Modeling
ISO	International Organization for Standardization
JBC	Joint Battle Center
JBMI	Joint Battle Management Initiative
JCDB	Joint Common Database
MIDB	Modernized Intelligence Database
MSXML	Microsoft XML Parser
NPS	Naval Postgraduate School
ODBC	Open Database Connectivity
OODB	Object-Oriented Database
OODBMS	Object Oriented Database Management System
OS-OTG	Over the Horizon Targeting Gold
R&R	Registry and Repository
RDBMS	Relational Database Management System
SAX	Simple API for XML
SAX	Simple API for XML
SGML	Standardized Generalized Markup Language
SHADE	SHARED Data Engineering
SME	Subject Matter Expert
SMEs	Subject Matter Experts
SQL	Structured Query Language
SQL	Structured Query Language
TARDEC	Tank-Automotive Research, Development and Engineering Center
TDBM	Track Database Manager
URI	Unified Resource Identifier

USMTF	United States Message Text Format
VMF	Variable Message Format
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

LIST OF REFERENCES

- [AMP99] Automated Message Preparation Course, FORSCOM, 1 November 1999
- [AND00] P. Anderson (and many others), Professional XML, WROX Press, 2000
- [BOUR00] R. Bourret, XML Namespaces FAQ, 2000
- [DISA00] S. Klynsma, Guidance on the Use of Extensible Markup Language within DoD, DISA, 29 August 2000
- [DISABRF] J Pipher, DIICOE Data Engineering XML Registry Brief, DISA, 26 February 1999
- [DISAWS] DISA DIICOE Web site,
<http://diides.ncr.disa.mil/shade/index.cfm>
- [DOD98] DoD 8320.1-M-1, Data Standardization Procedures, April 1998
- [EHL01] T. Ehrhardt and B. Lyttle, Interconnectivity Via a Consolidated Type Hierarchy and XML, NPS, December 2000
- [HIN00] D. Hina, Evaluation of the XML as a Means for Establishing Interoperability Between Heterogeneous DoD Databases, NPS, September 2000
- [HOP99] B. Hopkins, XML - A Tool for Interoperability, Logicon supporting Navy Center for Tactical Systems Interoperability, 8 June 1999
- [HZ01] H. Zobair, An Approach for Matching Corresponding Attributes in Legacy Heterogeneous DoD Databases, NPS/TACOM, June 2001
- [I3BR] GCCS I3 Brochure,
<http://c4iweb.spawar.navy.mil/pm/gccsi3>
- [IDEF93] Integration Definition for Information Modeling (IDEF1X), Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, 21 December 1993
- [INFWS] Informix Web Site
- [JBMI00] JBC, Joint Battle Management Initiative Assessment Plan Draft, 11 August 2000
- [JCDB99] R. Carnevale, The Joint Common Database, PEO C3S, 18 May 1999
- [JDD00] Data Dictionary for the Joint Common Database, PEO C3S Horizontal Technology Integration Office, 1 June 2000
- [MAP00] M. Cokus, XML-MTF Mapping Third Public Working Draft, MITRE Corp. supporting U.S. Air Force, 4 August 2000
- [MID98] Modernized Integrated Database Database Design Document V2.01, 29 June 1998
- [ODBCWP] Microsoft's ODBC Web Page,
www.microsoft.com/data/odbc
- [ORCWS] Oracle Web Site
- [PRXML00] Professional XML, Wrox Press, 2000

[RCWP] R. Cover, The XML Cover Pages - W3C DOM, 16
November 2000, www.oasis-open.org/cover/dom.html

[SEM99] W. Li, C. Clifton, SEMINT: A tool for identifying
attribute correspondences in heterogeneous
databases using neural networks, Data & Knowledge
Engineering, 23 February 1999

[SYBWS] Sybase Web Site

[W3CWP] World Wide Web Consortium Web Page:
<http://www.w3c.org>

[XMDB00] K. Williams, Professional XML Databases, Wrox
Press, 2000

[XMLIE99] A. Homer, XML IE5, Wrox Press, 1999

[XMLMP00] J. Schneider, XML-MTF Update Brief, MITRE Corp.
supporting U.S. Air Force, 29 May 2000

[XMLR1.0] XML Recommendation 1.0, W3C, 10 February 1998

[XMLV00] E. Masek, XML-VML Kickoff Brief, Mitre Inc., 2000

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..... 2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218
2. Dudley Knox Library..... 2
Naval Postgraduate School
411 Dyer Road
Monterey, California 93943-5101
3. Chairman, Code CS..... 1
Naval Postgraduate School
Monterey, California 93943-5100
4. Dr. Luqi, CS/Lq..... 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5100
5. Dr. Valdis Berzins..... 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5100
6. CAPT Paul Young..... 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5100
7. Mike Saboe, Associate Director..... 1
U.S. Army Next Generation Software Engineering
ATTN: AMSTA-TR-R/265
Warren, MI 48397-5000
8. Robert F. Halle..... 6
30455 Inkster
Franklin, Michigan 48025